经典计算机科学著作最新版

计算机程序设计艺术

第4卷 第0册 （双语版）

组合算法与布尔函数概论

The Art of Computer Programming, Volume 4
Introduction to Combinatorial Algorithms and Boolean Functions

Fascicle 0

黄林鹏 等译

（美）Donald E. Knuth 著

机械工业出版社

本书是《计算机程序设计艺术，第4卷：组合算法》的第0册。本册介绍了组合搜索的历史和演化，涉及组合搜索技术的理论和实践应用；探究了与布尔函数相关的所有重要问题，考察了如何最有效地计算一个布尔函数的值的技术。本册是《计算机程序设计艺术》第7章，即组合搜索这一长篇宏论的起始。

Simplified Chinese edition copyright © 2010 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *The Art of Computer Programming, Volume 4, Fascicle 0, Introduction to Combinatorial Algorithms and Boolean Functions* by Donald E. Knuth , Copyright © 2008.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley.

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

封底无防伪标均为盗版
版权所有，侵权必究
本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2009-1359

图书在版编目（CIP）数据

计算机程序设计艺术　第4卷　第0册：组合算法与布尔函数概论（双语版）/（美）克努特（Knuth, D. E.）著；黄林鹏译. —北京：机械工业出版社，2010.6

书名原文：The Art Of Computer Programming, Volume 4, Fascicle 0, Introduction to Combinatorial Algorithms and Boolean Functions

ISBN 978-7-111-30334-3

Ⅰ. 计…　Ⅱ. ① 克…　② 黄…　Ⅲ. 程序设计－英、汉　Ⅳ. TP311.1

中国版本图书馆CIP数据核字（2010）第063083号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：张少波
北京瑞德印刷有限公司印刷
2010年8月第1版第1次印刷
170mm × 242mm　. 28印张
标准书号：ISBN 978-7-111-30334-3
定价：69.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991，88361066
购书热线：（010）68326294，88379649，68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

PREFACE

To put all the good stuff into one book is patently impossible,
and attempting even to be reasonably comprehensive
about certain aspects of the subject is likely to lead to runaway growth.

— GERALD B. FOLLAND, "Editor's Corner" (2005)

La dernière chose qu'on trouve en faisant un ouvrage
est de savoir celle qu'il faut mettre la première.

— BLAISE PASCAL, *Pensées* 740 (c. 1660)

THIS BOOKLET is Fascicle 0 of *The Art of Computer Programming*, Volume 4: *Combinatorial Algorithms*. As explained in the preface to Fascicle 1 of Volume 1, I'm circulating the material in this preliminary form because I know that the task of completing Volume 4 will take many years; I can't wait for people to begin reading what I've written so far and to provide valuable feedback.

To put the material in context, this fascicle contains the opening sections intended to launch a long, long chapter on combinatorial algorithms. Chapter 7 is planned to be by far the longest single chapter of *The Art of Computer Programming*; it will eventually fill at least three volumes (namely Volumes 4A, 4B, and 4C), assuming that I'm able to remain healthy. Like the second-longest chapter (Chapter 5), it begins with pump-priming introductory material that comes before the main text, including dozens of exercises to get the ball rolling. A long voyage lies ahead, and some important provisions need to be brought on board before we embark. Furthermore I want to minimize the shock of transition between Chapter 6 and the new chapter, because Chapter 6 was originally written and published more than thirty years ago.

Chapter 7 proper begins with Section 7.1: Zeros and Ones, which is another sort of introduction, at a different level. It dives into the all-important topics that surround the study of Boolean functions, which essentially underly everything that computers do. Subsection 7.1.1, "Boolean basics," attempts to erect a solid foundation of theoretical and practical ideas on which we shall build significant superstructures later; subsection 7.1.2, "Boolean evaluation," considers how to compute Boolean functions with maximum efficiency.

The remaining parts of Section 7.1 — namely 7.1.3, "Bitwise tricks and techniques," and 7.1.4, "Binary decision diagrams" — will be published soon as Volume 4, Fascicle 1. Then comes Section 7.2, Generating All Possibilities; the fascicles for Section 7.2.1, "Generating basic combinatorial patterns," have already appeared in print. Section 7.2.2 will deal with backtracking in general.

And so it will go on, if all goes well; an outline of the entire Chapter 7 as currently envisaged appears on the `taocp` webpage that is cited on page ii.

These introductory sections have turned out to have more than twice as many exercises as I had originally planned. In fact, the total number of exercises in this fascicle (366) is almost unbelievable. But many of them are quite simple, intended to reinforce the reader's understanding of basic definitions, or to acquaint readers with the joys of *The Stanford GraphBase*. Other exercises were simply irresistible, as they cried out to be included here — although, believe it or not, I did reject more potential leads than I actually followed up.

I would like to express my indebtedness to the late Robert W Floyd, who made dozens of valuable suggestions when I asked him to look over the first draft of this material in 1977. Thanks also to Robin Wilson of the Open University for his careful reading and many detailed suggestions; and to hundreds of readers who provided fantastic feedback on early drafts that circulated on the Internet.

I shall happily pay a finder's fee of $2.56 for each error in this fascicle when it is first reported to me, whether that error be typographical, technical, or historical. The same reward holds for items that I forgot to put in the index. And valuable suggestions for improvements to the text are worth 32¢ each. (Furthermore, if you find a better solution to an exercise, I'll actually reward you with immortal glory instead of mere money, by publishing your name in the eventual book:–)

Notations that are used here and not otherwise explained can be found in the Index to Notations at the end of Volumes 1, 2, or 3. Those indexes point to the places where further information is available. (See also the entries under "Notation" in the present booklet.) Of course Volume 4 will some day contain its own Index to Notations.

Machine-language examples in all future editions of *The Art of Computer Programming* will be based on the `MMIX` computer, which is described in Volume 1, Fascicle 1.

Cross references to yet-unwritten material sometimes appear as '00' in the following pages; this impossible value is a placeholder for the actual numbers to be supplied later.

Happy reading!

Stanford, California D. E. K.
January 2008

PDG

PREFACE TO VOLUME 4

THE TITLE of Volume 4 is *Combinatorial Algorithms*, and when I proposed it I was strongly inclined to add a subtitle: *The Kind of Programming I Like Best*. My editors have decided to tone down such exuberance, but the fact remains that programs with a combinatorial flavor have always been my favorites.

On the other hand I've often been surprised to find that, in many people's minds, the word "combinatorial" is linked with computational difficulty. Indeed, Samuel Johnson, in his famous dictionary of the English language (1755), said that the corresponding noun "is now generally used in an ill sense." Colleagues tell me tales of woe, in which they report that "the combinatorics of the situation defeated us." Why is it that, for me, combinatorics arouses feelings of pure pleasure, yet for many others it evokes pure panic?

It's true that combinatorial problems are often associated with humongously large numbers. Johnson's dictionary entry also included a quote from Ephraim Chambers, who had stated that the total number of words of length 24 or less, in a 24-letter alphabet, is 1,391,724,288,887,252,999,425,128,493,402,200. The corresponding number for a 10-letter alphabet is 11,111,111,110; and it's only 3905 when the number of letters is 5. Thus a "combinatorial explosion" certainly does occur as the size of the alphabet grows from 5 to 10 to 24 and beyond.

Computing machines have become tremendously more powerful throughout my life. As I write these words, I know that they are being processed by a "laptop" whose speed is more than 100,000 times faster than the trusty IBM Type 650 computer to which I'm dedicating these books; my current machine's memory capacity is also more than 100,000 times greater. Tomorrow's computers will be even faster and more capacious. But these amazing advances have not diminished people's craving for answers to combinatorial questions; quite the contrary. Our once-unimaginable ability to compute so rapidly has raised our expectations, and whetted our appetite for more — because, in fact, the size of a combinatorial problem can increase more than 100,000-fold when n simply increases by 1.

Combinatorial algorithms can be defined informally as techniques for the high-speed manipulation of combinatorial objects such as permutations or graphs. We typically try to find patterns or arrangements that are the best possible ways to satisfy certain constraints. The number of such problems is vast, and the art of writing such programs is especially important and appealing because a single good idea can save years or even centuries of computer time.

Indeed, the fact that good algorithms for combinatorial problems can have a terrific payoff has led to terrific advances in the state of the art. Many problems that once were thought to be intractable can now be polished off with ease, and

many algorithms that once were known to be good have now become better. Starting about 1970, computer scientists began to experience a phenomenon that we called "Floyd's Lemma": Problems that seemed to need n^3 operations could actually be solved in $O(n^2)$; problems that seemed to require n^2 could be handled in $O(n \log n)$; and $n \log n$ was often reducible to $O(n)$. More difficult problems saw a reduction in running time from $O(2^n)$ to $O(1.5^n)$ to $O(1.3^n)$, etc. Other problems remained difficult in general, but they were found to have important special cases that are much simpler. Many combinatorial questions that I once thought would never be answered have now been resolved, and these breakthroughs are due mainly to improvements in algorithms rather than to improvements in processor speeds.

By 1975, such research was advancing so rapidly that a substantial fraction of the papers published in leading journals of computer science were devoted to combinatorial algorithms. And the advances weren't being made only by people in the core of computer science; significant contributions were coming from workers in electrical engineering, artificial intelligence, operations research, mathematics, physics, statistics, and other fields. I was trying to complete Volume 4 of *The Art of Computer Programming*, but instead I felt like I was sitting on the lid of a boiling kettle: I was confronted with a combinatorial explosion of another kind, a prodigious explosion of new ideas!

This series of books was born at the beginning of 1962, when I naïvely wrote out a list of tentative chapter titles for a 12-chapter book. At that time I decided to include a brief chapter about combinatorial algorithms, just for fun. "Hey look, most people use computers to deal with numbers, but we can also write programs that deal with patterns." In those days it was easy to give a fairly complete description of just about every combinatorial algorithm that was known. And even by 1966, when I'd finished a first draft of about 3000 handwritten pages for that already-overgrown book, fewer than 100 of those pages belonged to Chapter 7. I had absolutely no idea that what I'd foreseen as a sort of "salad course" would eventually turn out to be the main dish.

The great combinatorial fermentation of 1975 has continued to churn, as more and more people have begun to participate. New ideas improve upon the older ones, but rarely replace them or make them obsolete. So of course I've had to abandon any hopes that I once had of being able to surround the field, to write a definitive book that sets everything in order and provides one-stop shopping for everyone who has combinatorial problems to solve. It's almost never possible to discuss a subtopic and say, "Here's the final solution: end of story." Instead, I must restrict myself to explaining the most important principles that seem to underlie all of the efficient combinatorial methods that I've encountered so far. At present I've accumulated more than twice as much raw material for Volume 4 as for all of Volumes 1–3 combined.

This sheer mass of material implies that the once-planned "Volume 4" must actually become several physical volumes. You are now looking at Volume 4A. Volumes 4B and 4C will exist someday, assuming that I'm able to remain healthy; and (who knows?) there may also be Volumes 4D, 4E, . . . ; but surely not 4Z.

PDG

My plan is to go systematically through the files that I've amassed since 1962 and to tell the stories that I believe are still waiting to be told, to the best of my ability. I can't aspire to completeness, but I do want to give proper credit to all of the pioneers who have been responsible for key ideas; so I won't scrimp on historical details. Furthermore, whenever I learn something that I think is likely to remain important 50 years from now, something that can also be explained elegantly in a paragraph or two, I can't bear to leave it out. Conversely, difficult material that requires a lengthy proof is beyond the scope of these books, unless the subject matter is truly fundamental.

OK, it's clear that the field of Combinatorial Algorithms is vast, and I can't cover it all. What are the most important things that I'm leaving out? My biggest blind spot, I think, is geometry, because I've always been much better at visualizing and manipulating algebraic formulas than objects in space. Therefore I don't attempt to deal in these books with combinatorial problems that are related to computational geometry, such as close packing of spheres, or clustering of data points in n-dimensional Euclidean space, or even the Steiner tree problem in the plane. More significantly, I tend to shy away from polyhedral combinatorics, and from approaches that are based primarily on linear programming, integer programming, or semidefinite programming. Those topics are treated well in many other books on the subject, and they rely on geometrical intuition. Purely combinatorial developments are easier for me to understand.

I also must confess a bias against algorithms that are efficient only in an asymptotic sense, algorithms whose superior performance doesn't begin to "kick in" until the size of the problem exceeds the size of the universe. A great many publications nowadays are devoted to algorithms of that kind. I can understand why the contemplation of ultimate limits has intellectual appeal and carries an academic cachet; but in *The Art of Computer Programming* I tend to give short shrift to any methods that I would never consider using myself in an actual program. (There are, of course, exceptions to this rule, especially with respect to basic concepts in the core of the subject. Some impractical methods are simply too beautiful and/or too insightful to be excluded; others provide instructive examples of what *not* to do.)

Furthermore, as in earlier volumes of this series, I'm intentionally concentrating almost entirely on *sequential* algorithms, even though computers are increasingly able to carry out activities in parallel. I'm unable to judge what ideas about parallelism are likely to be useful five or ten years from now, let alone fifty, so I happily leave such questions to others who are wiser than I. Sequential methods, by themselves, already test the limits of my own ability to discern what the artful programmers of tomorrow will want to know.

The main decision that I needed to make when planning how to present this material was whether to organize it by problems or by techniques. Chapter 5 in Volume 3, for example, was devoted to a single problem, the sorting of data into order; more than two dozen techniques were applied to different aspects of that problem. Combinatorial algorithms, by contrast, involve many different problems, which tend to be attacked with a smaller repertoire of techniques.

I finally decided that a mixed strategy would work better than any pure approach. Thus, for example, these books treat the problem of finding shortest paths in Section 7.3, and problems of connectivity in Section 7.4.1; but many other sections are devoted to basic techniques, such as the use of Boolean algebra (Section 7.1), backtracking (Section 7.2), matroid theory (Section 7.6), or dynamic programming (Section 7.7). The famous Traveling Salesrep Problem, and other classic combinatorial tasks related to covering, coloring, and packing, have no sections of their own, but they come up several times in different places as they are treated by different methods.

I've mentioned great progress in the art of combinatorial computing, but I don't mean to imply that all combinatorial problems have actually been tamed. When the running time of a computer program goes ballistic, its programmers shouldn't expect to find a silver bullet for their needs in this book. The methods described here will often work a great deal faster than the first approaches that a programmer tries; but let's face it: Combinatorial problems get huge very quickly. We can even prove rigorously that a certain small, natural problem will *never* have a feasible solution in the real world, although it is solvable in principle (see the theorem of Stockmeyer and Meyer in Section 7.1.2). In other cases we cannot prove as yet that no decent algorithm for a given problem exists, but we know that such methods are unlikely, because any efficient algorithm would yield a good way to solve thousands of other problems that have stumped the world's greatest experts (see the discussion of NP-completeness in Section 7.9).

Experience suggests that new combinatorial algorithms will continue to be invented, for new combinatorial problems and for newly identified variations or special cases of old ones; and that people's appetite for such algorithms will also continue to grow. The art of computer programming continually reaches new heights when programmers are faced with challenges such as these. Yet today's methods are also likely to remain relevant.

Most of this book is self-contained, although there are frequent tie-ins with the topics discussed in Volumes 1–3. Low-level details of machine language programming have been covered extensively in those volumes, so the algorithms in the present book are usually specified only at an abstract level, independent of any machine. However, some aspects of combinatorial programming are heavily dependent on low-level details that didn't arise before; in such cases, all examples in this book are based on the `MMIX` computer, which supersedes the `MIX` machine that was defined in early editions of Volume 1. Details about `MMIX` appear in a paperback supplement to that volume called *The Art of Computer Programming*, Volume 1, Fascicle 1; they're also available on the Internet, together with downloadable assemblers and simulators.

Another downloadable resource, a collection of programs and data called *The Stanford GraphBase*, is cited extensively in the examples of this book. Readers are encouraged to play with it, in order to learn about combinatorial algorithms in what I think will be the most efficient and most enjoyable way.

Incidentally, while writing the introductory material at the beginning of Chapter 7, I was pleased to note that it was natural to mention some work of

my Ph.D. thesis advisor, Marshall Hall, Jr. (1910–1990), as well as some work of *his* thesis advisor, Oystein Ore (1899–1968), as well as some work of *his* thesis advisor, Thoralf Skolem (1887–1963). Skolem's advisor, Axel Thue (1863–1922), was already present in Chapter 6.

I'm immensely grateful to the hundreds of readers who have helped me to ferret out numerous mistakes that I made in early drafts of this volume, which were originally posted on the Internet and subsequently printed in paperback fascicles. But I fear that other errors still lurk among the details collected here, and I want to correct them as soon as possible. Therefore I will cheerfully pay $2.56 to the first finder of each technical, typographical, or historical error. The `taocp` webpage cited on page ii contains a current listing of all corrections that have been reported to me.

Stanford, California
April 2008

D. E. K.

Naturally, I am responsible for the remaining errors—
although, in my opinion, my friends could have caught a few more.
— CHRISTOS H. PAPADIMITRIOU, *Computational Complexity* (1995)

A note on references. References to *IEEE Transactions* include a letter code for the type of transactions, in boldface preceding the volume number. For example, '*IEEE Trans.* **C-35**' means the *IEEE Transactions on Computers*, volume 35. The IEEE no longer uses these convenient letter codes, but the codes aren't too hard to decipher: '**EC**' once stood for "Electronic Computers," '**IT**' for "Information Theory," '**SE**' for "Software Engineering," and '**SP**' for "Signal Processing," etc.; '**CAD**' meant "Computer-Aided Design of Integrated Circuits and Systems."

PDG

The author is especially grateful to the Addison–Wesley Publishing Company
for its patience in waiting a full decade for this manuscript
from the date the contract was signed.

— FRANK HARARY, *Graph Theory* (1968)

Bitte ein Bit!

— Slogan of Bitburger Brauerei (1951)

PDG

CONTENTS

目　录

PREFACE ······························ *iii*

PREFACE TO VOLUME 4 ············ *v*

Chapter 7　Combinatorial Searching

7.1　**Zeros and Ones** ·················· *47*

　　7.1.1　Boolean Basics ············ *47*

　　7.1.2　Boolean Evaluation ······ *96*

Answers to Exercises ················ *134*

Index and Glossary ················· *201*

译者序 ································ *219*

前言 ·································· *221*

第4卷前言 ···························· *223*

第7章　组 合 搜 索

7.1　0和1 ································ *274*

　　7.1.1　布尔基础 ················· *274*

　　7.1.2　布尔求值 ················· *321*

习题答案 ······························ *356*

CHAPTER SEVEN

COMBINATORIAL SEARCHING

You shall seeke all day ere you finde them,
& when you have them, they are not worth the search.
— BASSANIO, in *The Merchant of Venice* (Act I, Scene 1, Line 117)

Amid the action and reaction of so dense a swarm of humanity,
every possible combination of events may be expected to take place,
and many a little problem will be presented which may be striking and bizarre.
— SHERLOCK HOLMES, in *The Adventure of the Blue Carbuncle* (1892)

The field of combinatorial algorithms is too vast to cover
in a single paper or even in a single book.
— ROBERT E. TARJAN (1976)

While jostling against all manner of people
it has been impressed upon my mind that the successful ones
are those who have a natural faculty for solving puzzles.
Life is full of puzzles, and we are called upon
to solve such as fate throws our way.
— SAM LOYD, JR. (1927)

COMBINATORICS is the study of the ways in which discrete objects can be arranged into various kinds of patterns. For example, the objects might be $2n$ numbers $\{1, 1, 2, 2, \ldots, n, n\}$, and we might want to place them in a row so that exactly k numbers occur between the two appearances of each digit k. When $n = 3$ there is essentially only one way to arrange such "Langford pairs," namely 231213 (and its left-right reversal); similarly, there's also a unique solution when $n = 4$. Many other types of combinatorial patterns are discussed below.

Five basic types of questions typically arise when combinatorial problems are studied, some more difficult than others.

i) Existence: Are there any arrangements X that conform to the pattern?
ii) Construction: If so, can such an X be found quickly?
iii) Enumeration: How many different arrangements X exist?
iv) Generation: Can all arrangements X_1, X_2, … be visited systematically?
v) Optimization: What arrangements maximize or minimize $f(X)$, given an objective function f?

Each of these questions turns out to be interesting with respect to Langford pairs.

For example, consider the question of existence. Trial and error quickly reveals that, when $n = 5$, we cannot place $\{1,1,2,2,\ldots,5,5\}$ properly into ten positions. The two 1s must both go into even-numbered slots, or both into odd-numbered slots; similarly, the 3s and 5s must choose between two evens or two odds; but the 2s and 4s use one of each. Thus we can't fill exactly five slots of each parity. This reasoning also proves that the problem has no solution when $n = 6$, or in general whenever the number of odd values in $\{1,2,\ldots,n\}$ is odd.

In other words, Langford pairings can exist only when $n = 4m-1$ or $n = 4m$, for some integer m. Conversely, when n does have this form, Roy O. Davies has found an elegant way to construct a suitable placement (see exercise 1).

How many essentially different pairings, L_n, exist? Lots, when n grows:

$$\begin{aligned} L_3 &= 1; & L_4 &= 1;\\ L_7 &= 26; & L_8 &= 150;\\ L_{11} &= 17{,}792; & L_{12} &= 108{,}144;\\ L_{15} &= 39{,}809{,}640; & L_{16} &= 326{,}721{,}800;\\ L_{19} &= 256{,}814{,}891{,}280; & L_{20} &= 2{,}636{,}337{,}861{,}200;\\ L_{23} &= 3{,}799{,}455{,}942{,}515{,}488; & L_{24} &= 46{,}845{,}158{,}056{,}515{,}936. \end{aligned} \tag{1}$$

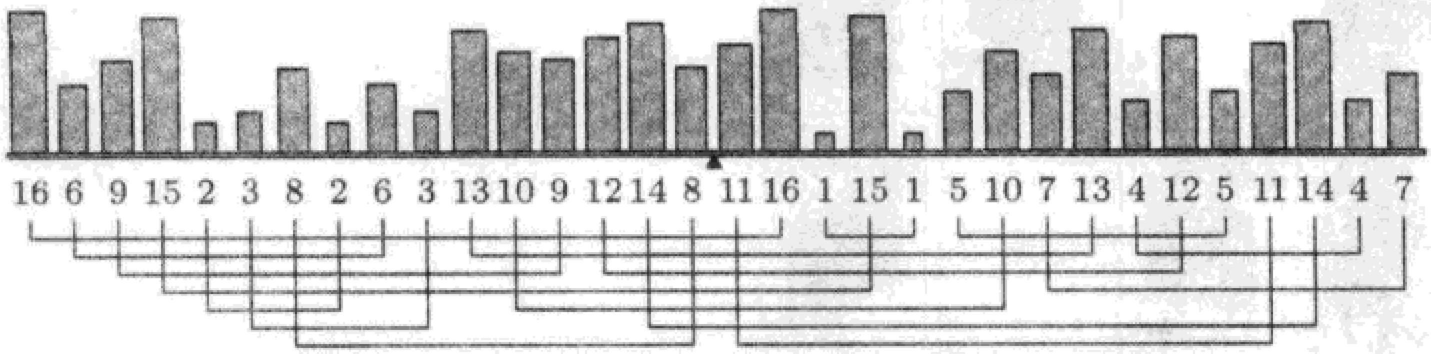
[The values of L_{23} and L_{24} were determined by M. Krajecki, C. Jaillet, and A. Bui in 2004 and 2005; see *Studia Informatica Universalis* **4** (2005), 151–190.] A seat-of-the-pants calculation suggests that L_n might be roughly of order $(4n/e^3)^{n+1/2}$ when it is nonzero (see exercise 5); and in fact this prediction turns out to be basically correct in all known cases. But no simple formula is apparent.

The problem of Langford arrangements is a simple special case of a general class of combinatorial challenges called *exact cover problems*. In Section 7.2.2.1 we shall study an algorithm called "dancing links," which is a convenient way to generate all solutions to such problems. When $n = 16$, for example, that method needs to perform only about 3200 memory accesses for each Langford pair arrangement that it finds. Thus the value of L_{16} can be computed in a reasonable amount of time by simply generating all of the pairings and counting them.

Notice, however, that L_{24} is a *huge* number — roughly 5×10^{16}, or about 1500 MIP-years. (Recall that a "MIP-year" is the number of instructions executed per year by a machine that carries out a million instructions per second, namely 31,556,952,000,000.) Therefore it's clear that the exact value of L_{24} was determined by some technique that did *not* involve generating all of the arrangements. Indeed, there is a much, much faster way to compute L_n, using polynomial algebra. The instructive method described in exercise 6 needs $O(4^n n)$ operations, which may seem inefficient; but it beats the generate-and-count method by a whopping factor of order $\Theta((n/e^3)^{n-1/2})$, and even when $n = 16$ it runs about 20 times faster. On the other hand, the exact value of L_{100} will probably never be known, even as computers become faster and faster.

We can also consider Langford pairings that are *optimum* in various ways. For example, it's possible to arrange sixteen pairs of weights $\{1,1,2,2,\ldots,16,16\}$ that satisfy Langford's condition and have the additional property of being "well-

balanced," in the sense that they won't tip a balance beam when they are placed in the appropriate order:



(2)

In other words, $15.5\cdot16+14.5\cdot6+\cdots+0.5\cdot8 = 0.5\cdot11+\cdots+14.5\cdot4+15.5\cdot7$; and in this particular example we also have another kind of balance, $16+6+\cdots+8 = 11+16+\cdots+7$, hence also $16\cdot16+15\cdot6+\cdots+1\cdot8 = 1\cdot11+\cdots+15\cdot4+16\cdot7$.

Moreover, the arrangement in (2) has *minimum width* among all Langford pairings of order 16: The connecting lines at the bottom of the diagram show that no more than seven pairs are incomplete at any point, as we read from left to right; and one can show that a width of six is impossible. (See exercise 7.)

What arrangements $a_1a_2\ldots a_{32}$ of $\{1,1,\ldots,16,16\}$ are the *least* balanced, in the sense that $\sum_{k=1}^{32} ka_k$ is maximized? The maximum possible value turns out to be 5268. One such pairing — there are 12,016 of them — is

$$2\;3\;4\;2\;1\;3\;1\;4\;16\;13\;15\;5\;14\;7\;9\;6\;11\;5\;12\;10\;8\;7\;6\;13\;9\;16\;15\;14\;11\;8\;10\;12. \qquad (3)$$

A more interesting question is to ask for the Langford pairings that are smallest and largest in lexicographic order. The answers for $n = 24$ are

$$\{\texttt{abacbdecfgdoersfpgqtuwxvjklonhmirpsjqkhltiunmwvx},\; \texttt{xvwsquntkigrdapaodgiknqsvxwutmrpohljcfbecbhmfejl}\} \qquad (4)$$

if we use the letters `a`, `b`, ..., `w`, `x` instead of the numbers 1, 2, ..., 23, 24.

We shall discuss many techniques for combinatorial optimization in later sections of this chapter. Our goal, of course, will be to solve such problems without examining more than a tiny portion of the space of all possible arrangements.

Orthogonal latin squares. Let's look back for a moment at the early days of combinatorics. A posthumous edition of Jacques Ozanam's *Recreations mathematiques et physiques* (Paris: 1725) included an amusing puzzle in volume 4, page 434: "Take all the aces, kings, queens, and jacks from an ordinary deck of playing cards and arrange them in a square so that each row and each column contains all four values and all four suits." Can you do it? Ozanam's solution, shown in Fig. 1 on the next page, does even more: It exhibits the full panoply of values and of suits also on both main diagonals. (Please don't turn the page until you've given this problem a try.)

By 1779 a similar puzzle was making the rounds of St. Petersburg, and it came to the attention of the great mathematician Leonhard Euler. "Thirty-six officers of six different ranks, taken from six different regiments, want to march in a 6×6 formation so that each row and each column will contain one officer of each rank and one of each regiment. How can they do it?" Nobody was able to

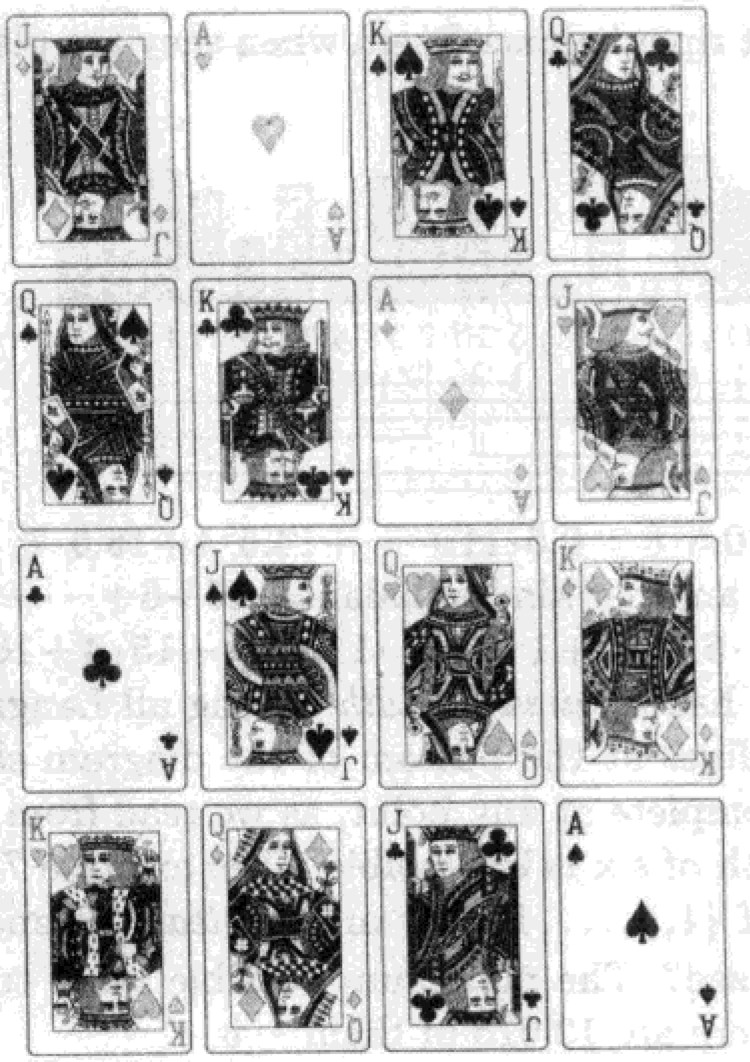


Fig. 1. Disorder in the court cards: No agreement in any line of four. (This configuration is one of many ways to solve a popular eighteenth-century problem.)

find a satisfactory marching order. So Euler decided to resolve the riddle — even though he had become nearly blind in 1771 and was dictating all of his work to assistants. He wrote a major paper on the subject [eventually published in *Verhandelingen uitgegeven door het Zeeuwsch Genootschap der Wetenschappen te Vlissingen* **9** (1782), 85–239], in which he constructed suitable arrangements for the analogous task with n ranks and n regiments when $n = 1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 16, \ldots$; only the cases with $n \bmod 4 = 2$ eluded him.

There's obviously no solution when $n = 2$. But Euler was stumped when $n = 6$, after having examined a "very considerable number" of square arrangements that didn't work. He showed that any actual solution would lead to many others that look different, and he couldn't believe that all such solutions had escaped his attention. Therefore he said, "I do not hesitate to conclude that one cannot produce a complete square of 36 cells, and that the same impossibility extends to the cases $n = 10$, $n = 14$ … in general to all oddly even numbers."

Euler named the 36 officers $a\alpha$, $a\beta$, $a\gamma$, $a\delta$, $a\epsilon$, $a\zeta$, $b\alpha$, $b\beta$, $b\gamma$, $b\delta$, $b\epsilon$, $b\zeta$, $c\alpha$, $c\beta$, $c\gamma$, $c\delta$, $c\epsilon$, $c\zeta$, $d\alpha$, $d\beta$, $d\gamma$, $d\delta$, $d\epsilon$, $d\zeta$, $e\alpha$, $e\beta$, $e\gamma$, $e\delta$, $e\epsilon$, $e\zeta$, $f\alpha$, $f\beta$, $f\gamma$, $f\delta$, $f\epsilon$, $f\zeta$, based on their regiments and ranks. He observed that any solution would amount to having two *separate* squares, one for Latin letters and another for Greek. Each of those squares is supposed to have distinct entries in rows and columns; so he began by studying the possible configurations for $\{a, b, c, d, e, f\}$, which he called *Latin squares*. A Latin square can be paired up with a Greek square to form a "Græco-Latin square" only if the squares are *orthogonal* to each other, meaning that no (Latin, Greek) pair of letters can be found together in more than one place when the squares are superimposed. For example, if we let $a = \mathtt{A}$, $b = \mathtt{K}$, $c = \mathtt{Q}$, $d = \mathtt{J}$, $\alpha = \clubsuit$, $\beta = \spadesuit$, $\gamma = \diamondsuit$, and $\delta = \heartsuit$, Fig. 1 is equivalent

to the Latin, Greek, and Græco-Latin squares

$$\begin{pmatrix} d & a & b & c \\ c & b & a & d \\ a & d & c & b \\ b & c & d & a \end{pmatrix}, \quad \begin{pmatrix} \gamma & \delta & \beta & \alpha \\ \beta & \alpha & \gamma & \delta \\ \alpha & \beta & \delta & \gamma \\ \delta & \gamma & \alpha & \beta \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} d\gamma & a\delta & b\beta & c\alpha \\ c\beta & b\alpha & a\gamma & d\delta \\ a\alpha & d\beta & c\delta & b\gamma \\ b\delta & c\gamma & d\alpha & a\beta \end{pmatrix}. \qquad (5)$$

Of course we can use *any* n distinct symbols in an $n \times n$ Latin square; all that matters is that no symbol occurs twice in any row or twice in any column. So we might as well use numeric values $\{0, 1, \ldots, n-1\}$ for the entries. Furthermore we'll just refer to "latin squares" (with a lowercase "l"), instead of categorizing a square as either Latin or Greek, because orthogonality is a symmetric relation.

Euler's assertion that two 6×6 latin squares cannot be orthogonal was verified by Thomas Clausen, who reduced the problem to an examination of 17 fundamentally different cases, according to a letter from H. C. Schumacher to C. F. Gauss dated 10 August 1842. But Clausen did not publish his analysis. The first demonstration to appear in print was by G. Tarry [*Comptes rendus, Association française pour l'avancement des sciences* **29**, part 2 (1901), 170–203], who discovered in his own way that 6×6 latin squares can be classified into 17 different families. (In Section 7.2.3 we shall study how to decompose a problem into combinatorially inequivalent classes of arrangements.)

Euler's conjecture about the remaining cases $n = 10$, $n = 14$, $\ldots$ was "proved" three times, by J. Petersen [*Annuaire des mathématiciens* (Paris: 1902), 413–427], by P. Wernicke [*Jahresbericht der Deutschen Math.-Vereinigung* **19** (1910), 264–267], and by H. F. MacNeish [*Annals of Math.* **23** (1922), 221–227]. Flaws in all three arguments became known, however; and the question was still unsettled when computers became available many years later. One of the very first combinatorial problems to be tackled by machine was therefore the enigma of 10×10 Græco-Latin squares: Do they exist or not?

In 1957, L. J. Paige and C. B. Tompkins programmed the SWAC computer to search for a counterexample to Euler's prediction. They selected one particular 10×10 latin square "almost at random," and their program tried to find another square that would be orthogonal to it. But the results were discouraging, and they decided to shut the machine off after five hours. Already the program had generated enough data for them to predict that at least 4.8×10^{11} hours of computer time would be needed to finish the run!

Shortly afterwards, three mathematicians made a breakthrough that put latin squares onto page one of major world newspapers: R. C. Bose, S. S. Shrikhande, and E. T. Parker found a remarkable series of constructions that yield orthogonal $n \times n$ squares for all $n > 6$ [*Proc. Nat. Acad. Sci.* **45** (1959), 734–737, 859–862; *Canadian J. Math.* **12** (1960), 189–203]. Thus, after resisting attacks for 180 years, Euler's conjecture turned out to be almost entirely wrong.

Their discovery was made without computer help. But Parker worked for UNIVAC, and he soon brought programming skills into the picture by solving the problem of Paige and Tompkins in less than an hour, on a UNIVAC 1206 Military Computer. [See *Proc. Symp. Applied Math.* **10** (1960), 71–83; **15** (1963), 73–81.]

Let's take a closer look at what the earlier programmers did, and how Parker dramatically trumped their approach. Paige and Tompkins began with the following 10×10 square L and its unknown orthogonal mate(s) M:

$$L = \begin{pmatrix} 0&1&2&3&4&5&6&7&8&9\\ 1&8&3&2&5&4&7&6&9&0\\ 2&9&5&6&3&0&8&4&7&1\\ 3&7&0&9&8&6&1&5&2&4\\ 4&6&7&5&2&9&0&8&1&3\\ 5&0&9&4&7&8&3&1&6&2\\ 6&5&4&7&1&3&2&9&0&8\\ 7&4&1&8&0&2&9&3&5&6\\ 8&3&6&0&9&1&5&2&4&7\\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \quad\text{and}\quad M = \begin{pmatrix} 0&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 1&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 2&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 3&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 4&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 5&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 6&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 7&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 8&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\ 9&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \end{pmatrix}. \qquad (6)$$

We can assume without loss of generality that the rows of M begin with 0, 1, ..., 9, as shown. The problem is to fill in the remaining 90 blank entries, and the original SWAC program proceeded from top to bottom, left to right. The top left $\sqcup$ can't be filled with 0, since 0 has already occurred in the top row of M. And it can't be 1 either, because the pair $(1, 1)$ already occurs at the left of the next row in (L, M). We can, however, tentatively insert a 2. The digit 1 can be placed next; and pretty soon we find the lexicographically smallest top row that might work for M, namely 0214365897. Similarly, the smallest rows that fit below 0214365897 are 1023456789 and 2108537946; and the smallest legitimate row below them is 3540619278. Now, unfortunately, the going gets tougher: There's no way to complete another row without coming into conflict with a previous choice. So we change 3540619278 to 3540629178 (but that doesn't work either), then to 3540698172, and so on for several more steps, until finally 3546109278 can be followed by 4397028651 before we get stuck again.

In Section 7.2.3, we'll study ways to estimate the behavior of such searches, without actually performing them. Such estimates tell us in this case that the Paige–Tompkins method essentially traverses an implicit search tree that contains about 2.5×10^{18} nodes. Most of those nodes belong to only a few levels of the tree; more than half of them deal with choices on the right half of the sixth row of M, after about 50 of the 90 blanks have been tentatively filled in. A typical node of the search tree probably requires about 75 mems (memory accesses) for processing, to check validity. Therefore the total running time on a modern computer would be roughly the time needed to perform 2×10^{20} mems.

Parker, on the other hand, went back to the method that Euler had originally used to search for orthogonal mates in 1779. First he found all of the so-called *transversals* of L, namely all ways to choose some of its elements so that there's exactly one element in each row, one in each column, and one of each value. For example, one transversal is 0859734216, in Euler's notation, meaning that we choose the 0 in column 0, the 8 in column 1, ..., the 6 in column 9. Each transversal that includes the k in L's leftmost column represents a legitimate way to place the ten k's into square M. The task of finding transversals is, in fact, rather easy, and the given matrix L turns out to have exactly 808 of them; there are respectively $(79, 96, 76, 87, 70, 84, 83, 75, 95, 63)$ transversals for $k = (0, 1, \ldots, 9)$.

Once the transversals are known, we're left with an exact cover problem of 10 stages, which is much simpler than the original 90-stage problem in (6). All we need to do is cover the square with ten transversals that don't intersect — because every such set of ten is equivalent to a latin square M that is orthogonal to L.

The particular square L in (6) has, in fact, exactly one orthogonal mate:

$$\begin{pmatrix} 0&1&2&3&4&5&6&7&8&9\\ 1&8&3&2&5&4&7&6&9&0\\ 2&9&5&6&3&0&8&4&7&1\\ 3&7&0&9&8&6&1&5&2&4\\ 4&6&7&5&2&9&0&8&1&3\\ 5&0&9&4&7&8&3&1&6&2\\ 6&5&4&7&1&3&2&9&0&8\\ 7&4&1&8&0&2&9&3&5&6\\ 8&3&6&0&9&1&5&2&4&7\\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \perp \begin{pmatrix} 0&2&8&5&9&4&7&3&6&1\\ 1&7&4&9&3&6&5&0&2&8\\ 2&5&6&4&8&7&0&1&9&3\\ 3&6&9&0&4&5&8&2&1&7\\ 4&8&1&7&5&3&6&9&0&2\\ 5&1&7&8&0&2&9&4&3&6\\ 6&9&0&2&7&1&3&8&4&5\\ 7&3&5&1&2&0&4&6&8&9\\ 8&0&2&3&6&9&1&7&5&4\\ 9&4&3&6&1&8&2&5&7&0 \end{pmatrix}. \qquad (7)$$

The dancing links algorithm finds it, and proves its uniqueness, after doing only about 1.7×10^8 mems of computation, given the 808 transversals. Furthermore, the cost of the transversal-finding phase, about 5 million mems, is negligible by comparison. Thus the original running time of 2×10^{20} mems — which once was regarded as the inevitable cost of solving a problem for which there are 10^{90} ways to fill in the blanks — has been reduced by a further factor of more than 10^{12}(!).

We will see later that advances have also been made in methods for solving 90-level problems like (6). Indeed, (6) turns out to be representable directly as an exact cover problem (see exercise 17), which the dancing links procedure of Section 7.2.2.1 solves after expending only 1.3×10^{11} mems. Even so, the Euler–Parker approach remains about a thousand times better than the Paige–Tompkins approach. By "factoring" the problem into two separate phases, one for transversal-finding and one for transversal-combining, Euler and Parker essentially reduced the computational cost from a product, T_1T_2, to a sum, T_1+T_2.

The moral of this story is clear: Combinatorial problems might confront us with a huge universe of possibilities, yet we shouldn't give up too easily. A single good idea can reduce the amount of computation by many orders of magnitude.

Puzzles versus the real world. Many of the combinatorial problems we shall study in this chapter, like Langford's problem of pairs or Ozanam's problem of the sixteen honor cards, originated as amusing puzzles or "brain twisters." Some readers might be put off by this emphasis on recreational topics, which they regard as a frivolous waste of time. Shouldn't computers really be doing useful work? And shouldn't textbooks about computers be primarily concerned with significant applications to industry and/or world progress?

Well, the author of the textbook you are reading has absolutely no objections to useful work and human progress. But he believes strongly that a book such as this should stress *methods* of problem solving, together with mathematical ideas and *models* that help to solve many different problems, rather than focusing on the reasons why those methods and models might be useful. We shall learn many beautiful and powerful ways to attack combinatorial problems, and the elegance

of those methods will be our main motivation for studying them. Combinatorial challenges pop up everywhere, and new ways to apply the techniques discussed in this chapter arise every day. So let's not limit our horizons by attempting to catalog in advance what the ideas are good for.

For example, it turns out that orthogonal latin squares are enormously useful, particularly in the design of experiments. Already in 1788, François Cretté de Palluel used a 4×4 latin square to study what happens when sixteen sheep — four each from four different breeds — were fed four different diets and harvested at four different times. [*Mémoires d'Agriculture* (Paris: Société Royale d'Agriculture, trimestre d'été, 1788), 17–23.] The latin square allowed him to do this with 16 sheep instead of 64; with a Græco-Latin square he could also have varied another parameter by trying, say, four different quantities of food or four different grazing paradigms.

But if we had focused our discussion on his approach to animal husbandry, we might well have gotten bogged down in details about breeding, about root vegetables versus grains and the costs of growing them, etc. Readers who aren't farmers might therefore have decided to skip the whole topic, even though latin square designs apply to a wide range of studies. (Think about testing five kinds of pills, on patients in five stages of some disease, five age brackets, and five weight groups.) Moreover, a concentration on experimental design could lead readers to miss the fact that latin squares also have important applications to coding and cryptography (see exercises 18–24).

Even the topic of Langford pairing, which seems at first to be purely recreational, turns out to have practical importance. T. Skolem used Langford sequences to construct Steiner triple systems, which we have applied to database queries in Section 6.5 [see *Math. Scandinavica* **6** (1958), 273–280]; and in the 1960s, E. J. Groth of Motorola Corporation applied Langford pairs to the design of circuits for multiplication. Furthermore, the algorithms that efficiently find Langford pairs and latin square transversals, such as the method of dancing links, apply to exact cover problems in general; and the problem of exact covering has great relevance to crucial problems such as the equitable apportionment of voter precincts to electoral districts, etc.

The applications are not the most important thing, and neither are the puzzles. Our primary goal is rather to get basic concepts into our brains, like the notions of latin squares and exact covering. Such notions give us the building blocks, vocabulary, and insights that *tomorrow's* problems will need.

Still, it's foolish to discuss problem solving without actually solving any problems. We need good problems to stimulate our creative juices, to light up our grey cells in a more or less organized fashion, and to make the basic concepts familiar. Mind-bending puzzles are often ideal for this purpose, because they can be presented in a few words, needing no complicated background knowledge.

Václav Havel once remarked that the complexities of life are vast: "There is too much to know... We have to abandon the arrogant belief that the world is merely a puzzle to be solved, a machine with instructions for use waiting to be discovered, a body of information to be fed into a computer." He called

for an increased sense of justice and responsibility; for taste, courage, and compassion. His words were filled with great wisdom. Yet thank goodness we do also have puzzles that *can* be solved! Puzzles deserve to be counted among the great pleasures of life, to be enjoyed in moderation like all other treats.

Of course, Langford and Ozanam directed their puzzles to human beings, not to computers. Aren't we missing the point if we merely shuffle such questions off to machines, to be solved by brute force instead of by rational thought? George Brewster, writing to Martin Gardner in 1963, expressed a widely held view as follows: "Feeding a recreational puzzle into a computer is no more than a step above dynamiting a trout stream. Succumbing to instant recreation."

Yes, but that view misses another important point: Simple puzzles often have generalizations that go beyond human ability and arouse our curiosity. The study of those generalizations often suggests instructive methods that apply to numerous other problems and have surprising consequences. Indeed, many of the key techniques that we shall study were born when people were trying to solve various puzzles. While writing this chapter, the author couldn't help relishing the fact that puzzles are now more fun than ever, as computers get faster and faster, because we keep getting more powerful dynamite to play with. [Further comments appear in the author's essay, "Can toy problems be useful?", originally written in 1976; see *Selected Papers on Computer Science* (1996), 169–183.]

Puzzles do have the danger that they can be *too* elegant. Good puzzles tend to be mathematically clean and well-structured, but we also need to learn how to deal systematically with the messy, chaotic, organic stuff that surrounds us every day. Indeed, some computational techniques are important chiefly because they provide powerful ways to cope with such complexities. That is why, for example, the arcane rules of library-card alphabetization were presented at the beginning of Chapter 5, and an actual elevator system was discussed at length to illustrate simulation techniques in Section 2.2.5.

A collection of programs and data called the Stanford GraphBase (SGB) has been prepared so that experiments with combinatorial algorithms can readily be performed on a variety of real-world examples. SGB includes, for example, data about American highways, and an input-output model of the U.S. economy; it records the casts of characters in Homer's *Iliad*, Tolstoy's *Anna Karenina*, and several other novels; it encapsulates the structure of Roget's *Thesaurus* of 1879; it documents hundreds of college football scores; it specifies the gray-value pixels of Leonardo da Vinci's *Gioconda* (Mona Lisa). And perhaps most importantly, SGB contains a collection of five-letter words, which we shall discuss next.

The five-letter words of English. Many of the examples in this chapter will be based on the following list of five-letter words:

$$\texttt{aargh}, \texttt{abaca}, \texttt{abaci}, \texttt{aback}, \texttt{abaft}, \texttt{abase}, \texttt{abash}, \ldots, \texttt{zooms}, \texttt{zowie}. \quad (8)$$

(There are 5757 words altogether — too many to display here; but those that are missing can readily be imagined.) It's a personal list, collected by the author between 1972 and 1992, beginning when he realized that such words would make **ideal** data for testing many kinds of combinatorial algorithms.

The list has intentionally been restricted to words that are **truly** part of the English language, in the sense that the author has encountered them in actual use. Unabridged dictionaries contain thousands of entries that are much more esoteric, like **aalii**, **abamp**, ..., **zymin**, and **zyxst**; words like that are useful primarily to Scrabble® players. But unfamiliar words tend to **spoil** the fun for anybody who doesn't know them. Therefore, for twenty years, the author systematically took note of all **words** that seemed **right** for the expository **goals** of *The Art of Computer Programming.*

Finally it was necessary to freeze the collection, in order to have a **fixed point** for reproducible experiments. The English language will always be evolving, but the 5757 SGB words will therefore always stay the same — even though the author has been tempted at times to add a few words that he didn't know in 1992, such as **chads**, **stent**, **blogs**, **ditzy**, **phish**, **bling**, and possibly **tetch**. No; **noway**. The time for any changes to SGB has long since **ended**: **finis**.

The following Glossary is intended to contain all well-known English words ... which may be used in good society, and which can serve as Links. ... There must be a stent to the admission of spick words.

— LEWIS CARROLL, *Doublets: A Word-Puzzle* (1879)

If there is such a verb as to tetch, Mr. Lillywaite tetched.

— ROBERT BARNARD, *Corpse in a Gilded Cage* (1984)

Proper names like **Knuth** are not considered to be legitimate words. But **gauss** and **hardy** are **valid**, because "gauss" is a unit of magnetic induction and "hardy" is hardy. In fact, SGB words are composed entirely of ordinary lowercase letters; the list contains no hyphenated words, contractions, or terms like **blasé** that require an accent. Thus each word can also be regarded as a vector, which has five components in the range $[0\,..\,26)$. In the vector sense, the words **yucca** and **abuzz** are furthest apart: The Euclidean distance between them is

$$\|(24,20,2,2,0)-(0,1,20,25,25)\|_2=\sqrt{24^2+19^2+18^2+23^2+25^2}=\sqrt{2415}.$$

The entire Stanford GraphBase, including all of its programs and data sets, is easy to download from the author's website (see page ii). And the list of all SGB words is even easier to obtain, because it is in the file '**sgb-words.txt**' at the same place. That file contains 5757 lines with one word per line, beginning with '**which**' and ending with '**pupal**'. The words appear in a default order, corresponding to frequency of usage; for example, the words of rank 1000, 2000, 3000, 4000, and 5000 are respectively **ditch**, **galls**, **visas**, **faker**, and **pismo**. The notation '$\mathtt{WORDS}(n)$' will be used in this chapter to stand for the n most common words, according to this ranking.

Incidentally, five-letter words include many plurals of *four-letter words*, and it should be noted that no Victorian-style censorship was done. Potentially offensive vocabulary has been expurgated from *The Official Scrabble® Players Dictionary*, but not from the SGB. One way to ensure that semantically unsuitable

terms will not appear in a professional paper based on the SGB wordlist is to restrict consideration to `WORDS`(n) where n is, say, 3000.

Exercises 26–37 below can be used as warmups for initial explorations of the SGB words, which we'll see in many different combinatorial contexts throughout this chapter. For example, while covering problems are still on our minds, we might as well note that the four words '**third flock began jumps**' cover 20 of the first 21 letters of the alphabet. Five words can, however, cover at most 24 different letters, as in {**becks**, **fjord**, **glitz**, **nymph**, **squaw**} — unless we resort to a rare non-SGB word like **waqfs** (Islamic endowments), which can be combined with {**gyved**, **bronx**, **chimp**, **klutz**} to cover 25.

Simple words from `WORDS`(400) suffice to make a *word square*:

```
class
light
agree .                                  (9)
sheep
steps
```

We need to go almost to `WORDS`(3000), however, to obtain a *word cube*,

```
types  yeast  pasta  ester  start
yeast  earth  armor  stove  three
pasta  armor  smoke  token  arena ,      (10)
ester  stove  token  event  rents
start  three  arena  rents  tease
```

in which every 5×5 "slice" is a word square. With a simple extension of the basic dancing links algorithm (see Section 7.2.2.2), one can show after performing about 390 billion mems of computation that `WORDS`(3000) supports only three symmetric word cubes such as (10); exercise 36 reveals the other two. Surprisingly, 83,576 symmetrical cubes can be made from the full set, `WORDS`(5757).

Graphs from words. It's interesting and important to arrange objects into rows, squares, cubes, and other designs; but in practical applications another kind of combinatorial structure is even *more* interesting and important, namely a *graph*. Recall from Section 2.3.4.1 that a graph is a set of points called *vertices*, together with a set of lines called *edges*, which connect certain pairs of vertices. Graphs are ubiquitous, and many beautiful graph algorithms have been discovered, so graphs will naturally be the primary focus of many sections in this chapter. In fact, the Stanford GraphBase is primarily about graphs, as its name implies; and the SGB words were collected chiefly because they can be used to define interesting and instructive graphs.

Lewis Carroll blazed the trail by inventing a game that he called Word-Links or Doublets, at the end of 1877. [See Martin Gardner, *The Universe in a Handkerchief* (1996), Chapter 6.] Carroll's idea, which soon became quite popular, was to transform one word to another by changing a letter at a time:

tears — **sears** — **stars** — **stare** — **stale** — **stile** — **smile**. (11)

The shortest such transformation is the shortest *path* in a graph, where the vertices of the graph are English words and the edges join pairs of words that have "Hamming distance 1" (meaning that they disagree in just one place).

When restricted to SGB words, Carroll's rule produces a graph of the Stanford GraphBase whose official name is *words*$(5757, 0, 0, 0)$. Every graph defined by SGB has a unique identifier called its *id*, and the graphs that are derived in Carrollian fashion from SGB words are identified by *ids* of the form *words*(n, l, t, s). Here n is the number of vertices; l is either 0 or a list of weights, used to emphasize various kinds of vocabulary; t is a threshold so that low-weight words can be disallowed; and s is the seed for any pseudorandom numbers that might be needed to break ties between words of equal weight. The full details needn't concern us, but a few examples will give the general idea:

- *words*$(n, 0, 0, 0)$ is precisely the graph that arises when Carroll's idea is applied to `WORDS`(n), for $1 \le n \le 5757$.
- *words*$(1000, \{0, 0, 0, 0, 0, 0, 0, 0, 0\}, 0, s)$ contains 1000 randomly chosen SGB words, usually different for different values of s.
- *words*$(766, \{0, 0, 0, 0, 0, 0, 0, 1, 0\}, 1, 0)$ contains all of the five-letter words that appear in the author's books about TEX and METAFONT.

There are only 766 words in the latter graph, so we can't form very many long paths like (11), although

$$\texttt{basic} — \texttt{basis} — \texttt{bases} — \texttt{based} — \texttt{baked} — \texttt{naked} — \texttt{named} — \texttt{names} — \texttt{games} \qquad (12)$$

is one noteworthy example.

Of course there are many other ways to define the edges of a graph when the vertices represent five-letter words. We could, for example, require the Euclidean distance to be small, instead of the Hamming distance. Or we could declare two words to be adjacent whenever they share a subword of length four; that strategy would substantially enrich the graph, making it possible for `chaos` to yield `peace`, even when confined to the 766 words that are related to TEX:

$$\texttt{chaos} — \texttt{chose} — \texttt{whose} — \texttt{whole} — \texttt{holes} — \texttt{hopes} — \texttt{copes} — \texttt{scope} — \texttt{score} — \texttt{store} — \texttt{stare} — \texttt{spare} — \texttt{space} — \texttt{paces} — \texttt{peace}. \qquad (13)$$

(In this rule we remove a letter, then insert another, possibly in a different place.) Or we might choose a totally different strategy, like putting an edge between word vectors $a_1a_2a_3a_4a_5$ and $b_1b_2b_3b_4b_5$ if and only if their dot product $a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5$ is a multiple of some parameter m. Graph algorithms thrive on different kinds of data.

SGB words lead also to an interesting family of *directed* graphs, if we write $a_1a_2a_3a_4a_5 \to b_1b_2b_3b_4b_5$ when $\{a_2, a_3, a_4, a_5\} \subseteq \{b_1, b_2, b_3, b_4, b_5\}$ as multisets. (Remove the first letter, insert another, and rearrange.) With this rule we can, for example, transform `words` to `graph` via a shortest oriented path of length six:

$$\texttt{words} \to \texttt{dross} \to \texttt{soars} \to \texttt{orcas} \to \texttt{crash} \to \texttt{sharp} \to \texttt{graph}. \qquad (14)$$

Theory is the first term in the Taylor series of practice.
— THOMAS M. COVER (1992)

The number of systems of terminology presently used in graph theory is equal, to a close approximation, to the number of graph theorists.
— RICHARD P. STANLEY (1986)

Graph theory: The basics. A graph G consists of a set V of vertices together with a set E of edges, which are pairs of distinct vertices. We will assume that V and E are *finite* sets unless otherwise specified. We write $u \mathrel{-\!\!-} v$ if u and v are vertices with $\{u,v\} \in E$, and $u \not\mathrel{-\!\!-} v$ if u and v are vertices with $\{u,v\} \notin E$. Vertices with $u \mathrel{-\!\!-} v$ are called "neighbors," and they're also said to be "adjacent" in G. One consequence of this definition is that we have $u \mathrel{-\!\!-} v$ if and only if $v \mathrel{-\!\!-} u$. Another consequence is that $v \not\mathrel{-\!\!-} v$, for all $v \in V$; that is, no vertex is adjacent to itself. (We shall, however, discuss multigraphs below, in which loops from a vertex to itself are permitted.)

The graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. It's a *spanning* subgraph of G if, in fact, $V' = V$. And it's an *induced* subgraph of G if E' has as many edges as possible, when V' is a given subset of the vertices. In other words, when $V' \subseteq V$ the subgraph of $G = (V, E)$ induced by V' is $G' = (V', E')$, where

$$E' \;=\; \bigl\{\,\{u,v\} \bigm| u \in V',\, v \in V',\, \text{and } \{u,v\} \in E\,\bigr\}. \tag{15}$$

This subgraph G' is denoted by $G \mid V'$, and often called "G restricted to V'." In the common case where $V' = V \setminus \{v\}$, we write simply $G \setminus v$ ("G minus vertex v") as an abbreviation for $G \mid (V \setminus \{v\})$. The similar notation $G \setminus e$ is used when $e \in E$ to denote the subgraph $G' = (V, E \setminus \{e\})$, obtained by removing an edge instead of a vertex. Notice that all of the SGB graphs known as *words*(n, l, t, s), described earlier, are induced subgraphs of the main graph *words*$(5757, 0, 0, 0)$; only the vocabulary changes in those graphs, not the rule for adjacency.

A graph with n vertices and e edges is said to have *order* n and *size* e. The simplest and most important graphs of order n are the *complete graph* K_n, the *path* P_n, and the *cycle* C_n. Suppose the vertices are $V = \{1, 2, \ldots, n\}$. Then

- K_n has $\binom{n}{2} = \frac{1}{2}n(n-1)$ edges $u \mathrel{-\!\!-} v$ for $1 \le u < v \le n$; every n-vertex graph is a spanning subgraph of K_n.
- P_n has $n-1$ edges $v \mathrel{-\!\!-} (v{+}1)$ for $1 \le v < n$, when $n \ge 1$; it is a path of length $n{-}1$ from 1 to n.
- C_n has n edges $v \mathrel{-\!\!-} ((v \bmod n){+}1)$ for $1 \le v \le n$; it is a graph only when $n = 0$ or $n \ge 3$ (but C_1 and C_2 are multigraphs).

We could actually have defined K_n, P_n, and C_n on the vertices $\{0, 1, \ldots, n{-}1\}$, or on *any* n-element set V instead of $\{1, 2, \ldots, n\}$, because two graphs that differ only in the names of their vertices but not in the structure of their edges are combinatorially equivalent.

Formally, we say that graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there is a one-to-one correspondence φ from V to V' such that $u \mathrel{-\!\!-} v$ in G if

and only if $\varphi(u) — \varphi(v)$ in G'. The notation $G \cong G'$ is often used to indicate that G and G' are isomorphic; but we shall often be less precise, by treating isomorphic graphs as if they were equal, and by occasionally writing $G = G'$ even when the vertex sets of G and G' aren't strictly identical.

Small graphs can be defined by simply drawing a diagram, in which the vertices are small circles and the edges are lines between them. Figure 2 illustrates several important examples, whose properties we will be studying later. The Petersen graph in Figure 2(e) is named after Julius Petersen, an early graph theorist who used it to disprove a plausible conjecture [L'*Intermédiaire des Mathématiciens* **5** (1898), 225–227]; it is, in fact, a remarkable configuration that serves as a counterexample to many optimistic predictions about what might be true for graphs in general. The Chvátal graph, Figure 2(f), was introduced by Václav Chvátal in *J. Combinatorial Theory* **9** (1970), 93–94.

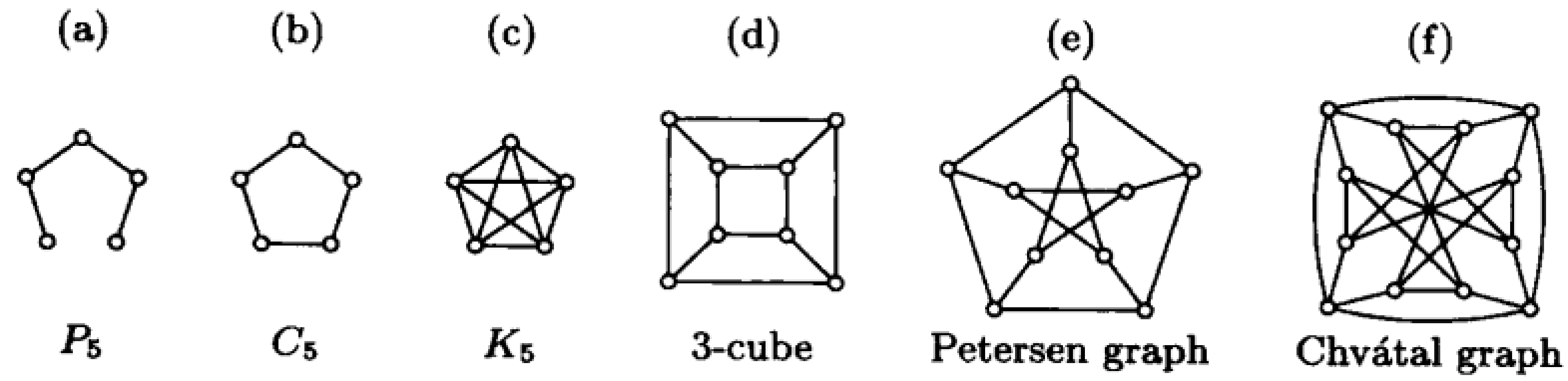
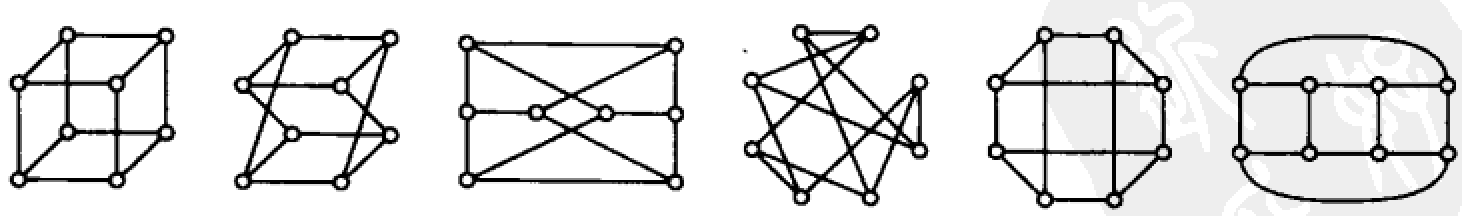


Fig. 2. Six example graphs, which have respectively (5, 5, 5, 8, 10, 12) vertices and (4, 5, 10, 12, 15, 24) edges.

The lines of a graph diagram are allowed to cross each other at points that aren't vertices. For example, the center point of Fig. 2(f) is *not* a vertex of Chvátal's graph. A graph is called *planar* if there's a way to draw it without any crossings. Clearly P_n and C_n are always planar; Fig. 2(d) shows that the 3-cube is also planar. But K_5 has too many edges to be planar (see exercise 46).

The *degree* of a vertex is the number of neighbors that it has. If all vertices have the same degree, the graph is said to be *regular*. In Fig. 2, for example, P_5 is irregular because it has two vertices of degree 1 and three of degree 2. But the other five graphs are regular, of degrees (2, 4, 3, 3, 4) respectively. A regular graph of degree 3 is often called "cubic" or "trivalent."

There are many ways to draw a given graph, some of which are much more perspicuous than others. For example, each of the six diagrams

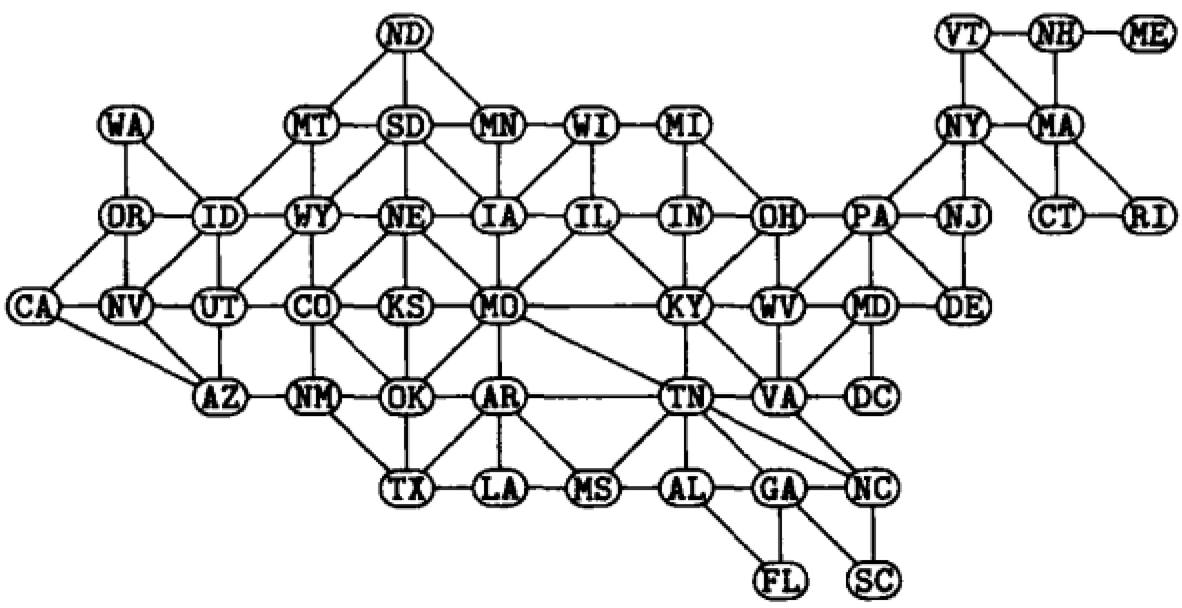

(16)

is isomorphic to the 3-cube, Fig. 2(d). The layout of Chvátal's graph that appears in Fig. 2(f) was discovered by Adrian Bondy many years after Chvátal's paper was published, thereby revealing unexpected symmetries.

The symmetries of a graph, also known as its *automorphisms*, are the permutations of its vertices that preserve adjacency. In other words, the permutation φ is an automorphism of G if we have $\varphi(u) — \varphi(v)$ whenever $u — v$ in G. A

well-chosen drawing like Fig. 2(f) can reveal underlying symmetry, but a single diagram isn't always able to display all the symmetries that exist. For example, the 3-cube has 48 automorphisms, and the Petersen graph has 120. We'll study algorithms that deal with isomorphisms and automorphisms in Section 7.2.3. Symmetries can often be exploited to avoid unnecessary computations, making an algorithm almost k times faster when it operates on a graph that has k automorphisms.

Graphs that have evolved in the real world tend to be rather different from the mathematically pristine graphs of Figure 2. For example, here's a familiar graph that has no symmetry whatsoever, although it does have the virtue of being planar:



(17)

It represents the contiguous United States of America, and we'll be using it later in several examples. The 49 vertices of this diagram have been labeled with two-letter postal codes for convenience, instead of being reduced to empty circles.

Paths and cycles. A spanning path of a graph is called a *Hamiltonian path*, and a spanning cycle is called a *Hamiltonian cycle*, because W. R. Hamilton invented and sold a puzzle in 1859 whose goal was to find such paths and cycles on the edges of a dodecahedron. T. P. Kirkman had independently studied the problem for polyhedra in general, in *Philosophical Transactions* **148** (1858), 145–161. [See *Graph Theory 1736–1936* by N. L. Biggs, E. K. Lloyd, and R. J. Wilson (1998), Chapter 2.] The task of finding a spanning path or cycle is, however, much older — indeed, we can legitimately consider it to be the oldest combinatorial problem of all, because paths and tours of a knight on a chessboard have a continuous history going back to ninth-century India (see Section 7.3.3). A graph is called *Hamiltonian* if it has a Hamiltonian cycle. (The Petersen graph, incidentally, is the smallest 3-regular graph that is neither planar nor Hamiltonian; see C. de Polignac, *Bull. Soc. Math. de France* **27** (1899), 142–145.)

The *girth* of a graph is the length of its shortest cycle; the girth is infinite if the graph is acyclic (containing no cycles). For example, the six graphs of Fig. 2 have girths $(\infty, 5, 3, 4, 5, 4)$, respectively. It's not difficult to prove that a graph of minimum degree k and girth 5 must have at least $k^2 + 1$ vertices. Further analysis shows in fact that this minimum value is achievable only if $k = 2$ (C_5), $k = 3$ (Petersen), $k = 7$, or perhaps $k = 57$. (See exercises 63 and 65.)

The *distance* $d(u,v)$ between two vertices u and v is the minimum length of a path from u to v in the graph; it is infinite if there's no such path. Clearly $d(v,v)=0$, and $d(u,v)=d(v,u)$. We also have the triangle inequality

$$d(u,v)+d(v,w) \;\ge\; d(u,w). \tag{18}$$

For if $d(u,v)=p$ and $d(v,w)=q$ and $p<\infty$ and $q<\infty$, there are paths

$$u=u_0 — u_1 — \cdots — u_p = v \quad \text{and} \quad v=v_0 — v_1 — \cdots — v_q = w, \tag{19}$$

and we can find the least subscript r such that $u_r = v_s$ for some s. Then

$$u_0 — u_1 — \cdots — u_{r-1} — v_s — v_{s+1} — \cdots — v_q \tag{20}$$

is a path of length $\le p+q$ from u to w.

The *diameter* of a graph is the maximum of $d(u,v)$, over all vertices u and v. The graph is *connected* if its diameter is finite. The vertices of a graph can always be partitioned into connected *components*, where two vertices u and v belong to the same component if and only if $d(u,v)<\infty$.

In the graph *words*$(5757,0,0,0)$, for example, we have $d(\texttt{tears},\texttt{smile})=6$, because (11) is a shortest path from `tears` to `smile`. Also $d(\texttt{tears},\texttt{happy})=6$, and $d(\texttt{smile},\texttt{happy})=10$, and $d(\texttt{world},\texttt{court})=6$. But $d(\texttt{world},\texttt{happy})=\infty$; the graph isn't connected. In fact, it contains 671 words like `aloof`, which have no neighbors and form connected components of order 1 all by themselves. Word pairs such as `alpha` — `aloha`, `droid` — `druid`, and `opium` — `odium` account for 103 further components of order 2. Some components of order 3, like `chain` — `chair` — `choir`, are paths; others, like $\{\texttt{getup},\texttt{letup},\texttt{setup}\}$, are cycles. A few more small components are also present, like the curious path

$$\texttt{login} — \texttt{logic} — \texttt{yogic} — \texttt{yogis} — \texttt{yogas} — \texttt{togas}, \tag{21}$$

whose words have no other neighbors. But the vast majority of all five-letter words belong to a giant component of order 4493. If you can go two steps away from a given word, the odds are better than 15 to 1 that your word is connected to everything in the giant component.

Similarly, the graph *words*$(n,0,0,0)$ has a giant component of order (3825, 2986, 2056, 1198, 224) when $n=(5000,4000,3000,2000,1000)$, respectively. But if n is small, there aren't enough edges to provide much connectivity. For example, *words*$(500,0,0,0)$ has 327 different components, none of order 15 or more.

The concept of distance can be generalized to $d(v_1,v_2,\ldots,v_k)$ for any value of k, meaning the minimum number of edges in a connected subgraph that contains the vertices $\{v_1,v_2,\ldots,v_k\}$. For example, $d(\texttt{blood},\texttt{sweat},\texttt{tears})$ turns out be 15, because the subgraph

$$\begin{array}{ccccccccccccc}
\texttt{blood} & — & \texttt{brood} & — & \texttt{broad} & — & \texttt{bread} & — & \texttt{tread} & — & \texttt{treed} & — & \texttt{tweed} \\
 & & & & & & & & & & | & & | \\
\texttt{tears} & — & \texttt{teams} & — & \texttt{trams} & — & \texttt{trims} & — & \texttt{tries} & — & \texttt{trees} & & \texttt{tweet} \\
 & & & & & & & & & & & & | \\
 & & & & & & & & & & \texttt{sweat} & — & \texttt{sweet}
\end{array} \tag{22}$$

has 15 edges, and there's no suitable 14-edge subgraph.

We noted in Section 2.3.4.1 that a connected graph with fewest edges is called a *free tree*. A subgraph that corresponds to the generalized distance $d(v_1, \ldots, v_k)$ will always be a free tree. It is misleadingly called a *Steiner tree*, because Jacob Steiner once mentioned the case $k = 3$ for points $\{v_1, v_2, v_3\}$ in the Euclidean plane [*Crelle* **13** (1835), 362–363]. Franz Heinen had solved that problem in *Über Systeme von Kräften* (1834); Gauss extended the analysis to $k = 4$ in a letter to Schumacher (21 March 1836).

Coloring. A graph is said to be *k-partite* or *k-colorable* if its vertices can be partitioned into k or fewer parts, with the endpoints of each edge belonging to different parts — or equivalently, if there's a way to paint its vertices with at most k different colors, never assigning the same color to two adjacent vertices. The famous Four Color Theorem, conjectured by F. Guthrie in 1852 and finally proved with massive computer aid by K. Appel, W. Haken, and J. Koch [*Illinois J. Math.* **21** (1977), 429–567], states that *every planar graph is 4-colorable.* No simple proof is known, but special cases like (17) can be colored at sight (see exercise 45); and $O(n^2)$ steps suffice to 4-color a planar graph in general [N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas, *STOC* **28** (1996), 571–575].

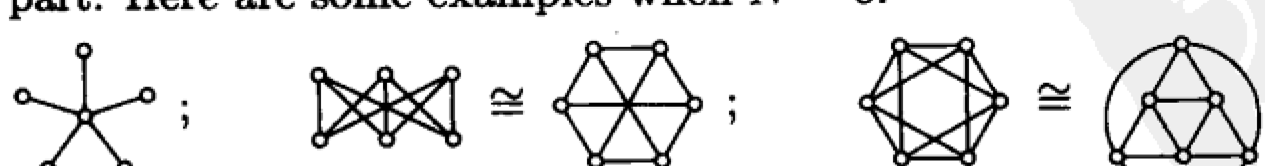
The case of 2-colorable graphs is especially important in practice. A 2-partite graph is generally called *bipartite*, or simply a "bigraph"; every edge of such a graph has one endpoint in each part.

Theorem B. *A graph is bipartite if and only if it contains no cycle of odd length.*

Proof. [See D. König, *Math. Annalen* **77** (1916), 453–454.] Every subgraph of a k-partite graph is k-partite. Therefore the cycle C_n can be a subgraph of a bipartite graph only if C_n itself is a bigraph, in which case n must be even.

Conversely, if a graph contains no odd cycles we can color its vertices with the two colors $\{0, 1\}$ by carrying out the following procedure: Begin with all vertices uncolored. If all neighbors of colored vertices are already colored, choose an uncolored vertex w, and color it 0. Otherwise choose a colored vertex u that has an uncolored neighbor v; assign to v the opposite color. Exercise 48 proves that a valid 2-coloring is eventually obtained. ∎

The *complete bipartite graph* $K_{m,n}$ is the largest bipartite graph whose vertices have two parts of sizes m and n. We can define it on the vertex set $\{1, 2, \ldots, m+n\}$ by saying that $u — v$ whenever $1 \le u \le m < v \le m + n$. In other words, $K_{m,n}$ has mn edges, one for each way to choose one vertex in the first part and another in the second part. Similarly, the *complete k-partite graph* $K_{n_1,\ldots,n_k}$ has $N = n_1 + \cdots + n_k$ vertices partitioned into parts of sizes $\{n_1, \ldots, n_k\}$, and it has edges between any two vertices that don't belong to the same part. Here are some examples when $N = 6$:



$K_{1,5}$; $K_{3,3}$; $K_{2,2,2}$. (23)

Notice that $K_{1,n}$ is a free tree; it is popularly called the *star graph* of order $n+1$.

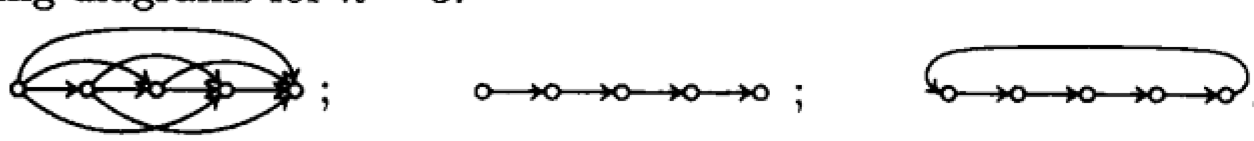
From now on say "digraph" instead of "directed graph."
It is clear and short and it will catch on.
— GEORGE PÓLYA, letter to Frank Harary (c. 1954)

Directed graphs. In Section 2.3.4.2 we defined *directed graphs* (or *digraphs*), which are very much like graphs except that they have *arcs* instead of edges. An arc $u \longrightarrow v$ runs from one vertex to another, while an edge $u \;—\; v$ joins two vertices without distinguishing between them. Furthermore, digraphs are allowed to have self-loops $v \longrightarrow v$ from a vertex to itself, and more than one arc $u \longrightarrow v$ may be present between the same vertices u and v.

Formally, a digraph $D = (V, A)$ of order n and size m is a set V of n vertices and a multiset A of m ordered pairs (u, v), where $u \in V$ and $v \in V$. The ordered pairs are called arcs, and we write $u \longrightarrow v$ when $(u, v) \in A$. The digraph is called *simple* if A is actually a set instead of a general multiset — namely, if there's at most one arc (u, v) for all u and v. Each arc (u, v) has an initial vertex u and a final vertex v, also called its "tip." Each vertex has an *out-degree* $d^+(v)$, the number of arcs for which v is the initial vertex, and an *in-degree* $d^-(v)$, the number of arcs for which v is the tip. A vertex with in-degree 0 is called a "source"; a vertex with out-degree 0 is called a "sink." Notice that $\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v)$, because both sums are equal to m, the total number of arcs.

Most of the notions we've defined for graphs carry over to digraphs in a natural way, if we just insert the word "directed" or "oriented" (or the syllable "di") when it's necessary to distinguish between edges and arcs. For example, digraphs have subdigraphs, which can be spanning or induced or neither. An isomorphism between digraphs $D = (V, A)$ and $D' = (V', A')$ is a one-to-one correspondence φ from V to V' for which the number of arcs $u \longrightarrow v$ in D equals the number of arcs $\varphi(u) \longrightarrow \varphi(v)$ in D', for all $u, v \in V$.

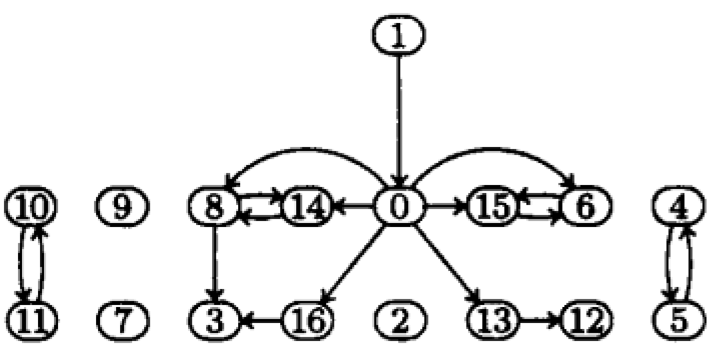
Diagrams for digraphs use arrows between the vertices, instead of unadorned lines. The simplest and most important digraphs of order n are directed variants of the graphs K_n, P_n, and C_n, namely the *transitive tournament* $K_n^{\rightarrow}$, the *oriented path* $P_n^{\rightarrow}$, and the *oriented cycle* $C_n^{\rightarrow}$. They can be schematically indicated by the following diagrams for $n = 5$:



$K_5^{\rightarrow}$; $P_5^{\rightarrow}$; $C_5^{\rightarrow}$. (24)

There's also the *complete digraph* J_n, which is the largest simple digraph on n vertices; it has n^2 arcs $u \longrightarrow v$, one for each choice of u and v.

Figure 3 shows a more elaborate diagram, for a digraph of order 17 that we might call "expressly oriented": It is the directed graph described by Hercule Poirot in Agatha Christie's novel *Murder on the Orient Express* (1934). Vertices correspond to the berths of the Stamboul–Calais coach in that story, and an arc $u \longrightarrow v$ means that the occupant of berth u has corroborated the alibi of the person in berth v. This example has six connected components, namely $\{0, 1, 3, 6, 8, 12, 13, 14, 15, 16\}$, $\{2\}$, $\{4, 5\}$, $\{7\}$, $\{9\}$, and $\{10, 11\}$, because connectivity in a digraph is determined by treating arcs as edges.



LEGEND
0: Pierre Michel, the French conductor
1: Hercule Poirot, the Belgian detective
2: Samuel Edward Ratchett, the deceased American
3: Caroline Martha Hubbard, the American matron
4: Edward Henry Masterman, the British valet
5: Antonio Foscarelli, the Italian automobile salesman
6: Hector MacQueen, the American secretary
7: Harvey Harris, the Englishman who didn't show up
8: Hildegarde Schmidt, the German lady's maid
9: (vacancy)
10: Greta Ohlsson, the Swedish nurse
11: Mary Hermione Debenham, the English governess
12: Helena Maria Andrenyi, the beautiful countess
13: Rudolph Andrenyi, the Hungarian count/diplomat
14: Natalia Dragomiroff, the Russian princess dowager
15: Colonel Arbuthnot, the British officer from India
16: Cyrus Bettman Hardman, the American detective

Fig. 3. A digraph of order 17 and size 18, devised by Agatha Christie.

Two arcs are *consecutive* if the tip of the first is the initial vertex of the second. A sequence of consecutive arcs $(a_1, a_2, \ldots, a_k)$ is called a *walk* of length k; it can be symbolized by showing the vertices as well as the arcs:

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \cdots v_{k-1} \xrightarrow{a_k} v_k. \qquad (25)$$

In a simple digraph it's sufficient merely to specify the vertices; for example, $1 \longrightarrow 0 \longrightarrow 8 \longrightarrow 14 \longrightarrow 8 \longrightarrow 3$ is a walk in Fig. 3. The walk in (25) is an oriented path when the vertices $\{v_0, v_1, \ldots, v_k\}$ are distinct; it's an oriented cycle when they are distinct except that $v_k = v_0$.

In a digraph, the directed distance $d(u, v)$ is the number of arcs in the shortest *oriented* path from u to v, which is also the length of the shortest walk from u to v. It may differ from $d(v, u)$; but the triangle inequality (18) remains valid.

Every graph can be regarded as a digraph, because an edge $u — v$ is essentially equivalent to a matched pair of arcs, $u \longrightarrow v$ and $v \longrightarrow u$. The digraph obtained in this way retains all the properties of the original graph; for example, the degree of each vertex in the graph becomes its out-degree in the digraph, and also its in-degree in the digraph. Furthermore, distances remain the same.

A *multigraph* (V, E) is like a graph except that its edges E can be any *multiset* of pairs $\{u, v\}$; edges $v — v$ that loop from a vertex to itself, which correspond to "multipairs" $\{v, v\}$, are also permitted. For example,



(26)

is a multigraph of order 3 with six edges, $\{1,1\}$, $\{1,2\}$, $\{2,3\}$, $\{2,3\}$, $\{3,3\}$, and $\{3,3\}$. The vertex degrees in this example are $d(1) = d(2) = 3$ and $d(3) = 6$, because each loop contributes 2 to the degree of its vertex. An edge loop $v — v$ becomes *two* arc loops $v \longrightarrow v$ when a multigraph is regarded as a digraph.

Representation of graphs and digraphs. Any digraph, and therefore any graph or multigraph, is completely described by its *adjacency matrix* $A = (a_{uv})$, which has n rows and n columns when there are n vertices. Each entry a_{uv} of this matrix specifies the number of arcs from u to v. For example, the adjacency matrices for $\vec{K_3}$, $\vec{P_3}$, $\vec{C_3}$, J_3, and (26) are respectively

$$\vec{K_3} = \begin{pmatrix} 011 \\ 001 \\ 000 \end{pmatrix}, \quad \vec{P_3} = \begin{pmatrix} 010 \\ 001 \\ 000 \end{pmatrix}, \quad \vec{C_3} = \begin{pmatrix} 010 \\ 001 \\ 100 \end{pmatrix}, \quad J_3 = \begin{pmatrix} 111 \\ 111 \\ 111 \end{pmatrix}, \quad A = \begin{pmatrix} 210 \\ 102 \\ 024 \end{pmatrix}. \qquad (27)$$

The powerful mathematical tools of matrix theory make it possible to prove many nontrivial results about graphs by studying their adjacency matrices; exercise 65 provides a particularly striking example of what can be done. One of the main reasons is that matrix multiplication has a simple interpretation in the context of digraphs. Consider the square of A, where the element in row u and column v is

$$(A^2)_{uv} \;=\; \sum_{w\in V} a_{uw}a_{wv}, \tag{28}$$

by definition. Since a_{uw} is the number of arcs from u to w, we see that $a_{uw}a_{wv}$ is the number of walks of the form $u \longrightarrow w \longrightarrow v$. Therefore $(A^2)_{uv}$ is the total number of walks of length 2 from u to v. Similarly, the entries of A^k tell us the total number of walks of length k between any ordered pair of vertices, for all $k \geq 0$. For example, the matrix A in (27) satisfies

$$A=\begin{pmatrix}2&1&0\\1&0&2\\0&2&4\end{pmatrix},\qquad A^2=\begin{pmatrix}5&2&2\\2&5&8\\2&8&20\end{pmatrix},\qquad A^3=\begin{pmatrix}12&9&12\\9&18&42\\12&42&96\end{pmatrix}; \tag{29}$$

there are 12 walks of length 3 from the vertex 1 of the multigraph (26) to vertex 3, and 18 such walks from vertex 2 to itself.

Reordering of the vertices changes an adjacency matrix from A to $P^{-}AP$, where P is a permutation matrix (a 0–1 matrix with exactly one 1 in each row and column), and $P^{-} = P^T$ is the matrix for the inverse permutation. Thus

$$\begin{pmatrix}210\\102\\024\end{pmatrix},\quad \begin{pmatrix}201\\042\\120\end{pmatrix},\quad \begin{pmatrix}012\\120\\204\end{pmatrix},\quad \begin{pmatrix}021\\240\\102\end{pmatrix},\quad \begin{pmatrix}402\\021\\210\end{pmatrix},\quad \text{and}\quad \begin{pmatrix}420\\201\\012\end{pmatrix} \tag{30}$$

are all adjacency matrices for (26), and there are no others.

There are more than $2^{n(n-1)/2}/n!$ graphs of order n, when $n > 1$, and almost all of them require $\Omega(n^2)$ bits of data in their most economical encoding. Consequently the best way to represent the vast majority of all possible graphs inside a computer, from the standpoint of memory usage, is essentially to work with their adjacency matrices.

But the graphs that actually arise in practical problems have quite different characteristics from graphs that are chosen at random from the set of all possibilities. A real-life graph usually turns out to be "sparse," having say $O(n \log n)$ edges instead of $\Omega(n^2)$, unless n is rather small, because $\Omega(n^2)$ bits of data are difficult to generate. For example, suppose the vertices correspond to people, and the edges correspond to friendships. If we consider 5 billion people, few of them will have more than 10000 friends. But even if everybody had 10000 friends, on average, the graph would still have only 2.5×10^{13} edges, while almost all graphs of order 5 billion have approximately 6.25×10^{18} edges.

Thus the best way to represent a graph inside a machine usually turns out to be rather different than to record n^2 values a_{uv} of adjacency matrix elements. Instead, the algorithms of the Stanford GraphBase were developed with a data structure akin to the linked representation of sparse matrices discussed in Section 2.2.6, though somewhat simplified. That approach has proved to be not only versatile and efficient, but also easy to use.

The SGB representation of a digraph is a combination of sequential and linked allocation, using nodes of two basic types. Some nodes represent vertices, other nodes represent arcs. (There's also a third type of node, which represents an entire graph, for algorithms that deal with several graphs at once. But each graph needs only one graph node, so the vertex and arc nodes predominate.)

Here's how it works: Every SGB digraph of order n and size m is built upon a sequential array of n vertex nodes, making it easy to access vertex k for $0 \leq k < n$. The m arc nodes, by contrast, are linked together within a general memory pool that is essentially unstructured. Each vertex node typically occupies 32 bytes, and each arc node occupies 20 (and the graph node occupies 220); but the node sizes can be modified without difficulty. A few fields of each node have a fixed, definite meaning in all cases; the remaining fields can be used for different purposes in different algorithms or in different phases of a single algorithm. The fixed-purpose parts of a node are called its "standard fields," and the multipurpose parts are called its "utility fields."

Every vertex node has two standard fields called `NAME` and `ARCS`. If v is a variable that points to a vertex node, we'll call it a *vertex variable*. Then `NAME`(v) points to a string of characters that can be used to identify the corresponding vertex in human-oriented output; for example, the 49 vertices of graph (17) have names like `CA`, `WA`, `OR`, …, `RI`. The other standard field, `ARCS`(v), is far more important in algorithms: It points to an arc node, the first in a singly linked list of length $d^+(v)$, with one node for each arc that emanates from vertex v.

Every arc node has two standard fields called `TIP` and `NEXT`; a variable a that points to an arc node is called an *arc variable*. `TIP`(a) points to the vertex node that represents the tip of arc a; `NEXT`(a) points to the arc node that represents the next arc whose initial vertex agrees with that of a.

A vertex v with out-degree 0 is represented by letting `ARCS`$(v) = \Lambda$ (the null pointer). Otherwise if, say, the out-degree is 3, the data structure contains three arc nodes with `ARCS`$(v) = a_1$, `NEXT`$(a_1) = a_2$, `NEXT`$(a_2) = a_3$, and `NEXT`$(a_3) = \Lambda$; and the three arcs from v lead to `TIP`(a_1), `TIP`(a_2), `TIP`(a_3).

Suppose, for example, that we want to compute the out-degree of vertex v, and store it in a utility field called `ODEG`. It's easy:

$$\begin{aligned}&\text{Set } a \leftarrow \texttt{ARCS}(v) \text{ and } d \leftarrow 0.\\&\text{While } a \neq \Lambda, \text{ set } d \leftarrow d+1 \text{ and } a \leftarrow \texttt{NEXT}(a).\\&\text{Set } \texttt{ODEG}(v) \leftarrow d.\end{aligned} \qquad (31)$$

When a graph or a multigraph is considered to be a digraph, as mentioned above, its edges $u - v$ are each equivalent to two arcs, $u \to v$ and $v \to u$. These arcs are called "mates"; and they occupy two arc nodes, say a and a', where a appears in the list of arcs from u and a' appears in the list of arcs from v. Then `TIP`$(a) = v$ and `TIP`$(a') = u$. We'll also write

$$\texttt{MATE}(a) = a' \qquad \text{and} \qquad \texttt{MATE}(a') = a, \qquad (32)$$

in algorithms that want to move rapidly from one list to another. However, we usually won't need to store an explicit pointer from an arc to its mate, or to have

a utility field called MATE within each arc node, because the necessary link can be deduced *implicitly* when the data structure has been constructed cleverly.

The implicit-mate trick works like this: While creating each edge $u — v$ of an undirected graph or multigraph, we introduce *consecutive* arc nodes for $u \longrightarrow v$ and $v \longrightarrow u$. For example, if there are 20 bytes per arc node, we'll reserve 40 consecutive bytes for each new pair. We can also make sure that the memory address of the first byte is a multiple of 8. Then if the arc node a is in memory location α, its mate is in location

$$\begin{Bmatrix} \alpha + 20, & \text{if } \alpha \bmod 8 = 0 \\ \alpha - 20, & \text{if } \alpha \bmod 8 = 4 \end{Bmatrix} = \alpha - 20 + \bigl(40 \mathbin{\&} ((\alpha \mathbin{\&} 4) - 1)\bigr). \tag{33}$$

Such tricks are valuable in combinatorial problems, when operations might be performed a trillion times, because every way to save 3.6 nanoseconds per operation will make such a computation finish an hour sooner. But (33) isn't directly "portable" from one implementation to another. If the size of an arc node were changed from 20 to 24, for example, we would have to change the numbers 40, 20, 8, and 4 in (33) to 48, 24, 16, and 8.

The algorithms in this book will make no assumptions about node sizes. Instead, we'll adopt a convention of the C programming language and its descendants, so that if a points to an arc node, '$a+1$' denotes a pointer to the arc node that follows it in memory. And in general

$$\mathtt{LOC(NODE}(a+k)\mathtt{))} = \mathtt{LOC(NODE}(a)\mathtt{))} + kc, \tag{34}$$

when there are c bytes in each arc node. Similarly, if v is a vertex variable, '$v+k$' will stand for the kth vertex node following node v; the actual memory location of that node will be v plus k times the size of a vertex node.

The standard fields of a graph node g include M(g), the total number of arcs; N(g), the total number of vertices; VERTICES(g), a pointer to the first vertex node in the sequential list of all vertex nodes; ID(g), the graph's identification, which is a string like words(5757,0,0,0); and some other fields needed for the allocation and recycling of memory when the graph grows or shrinks, or for exporting a graph to external formats that interface with other users and other graph-manipulation systems. But we will rarely need to refer to any of these graph node fields, nor will it be necessary to give a complete description of SGB format here, since we shall describe almost all of the graph algorithms in this chapter by sticking to an English-language description at a fairly abstract level instead of descending to the bit level of machine programs.

A simple graph algorithm. To illustrate a medium-high-level algorithm of the kind that will appear later, let's convert the proof of Theorem B into a step-by-step procedure that paints the vertices of a given graph with two colors whenever that graph is bipartite.

Algorithm B (*Bipartiteness testing*). Given a graph represented in SGB format, this algorithm either finds a 2-coloring with COLOR(v) $\in \{0,1\}$ in each vertex v, or it terminates unsuccessfully when no valid 2-coloring is possible. Here COLOR is a utility field in each vertex node. Another vertex utility field, LINK(v), is a

vertex pointer used to maintain a stack of all colored vertices whose neighbors have not yet been examined. An auxiliary vertex variable s points to the top of this stack. The algorithm also uses variables u, v, w for vertices and a for arcs. The vertex nodes are assumed to be $v_0 + k$ for $0 \le k < n$.

B1. [Initialize.] Set $\texttt{COLOR}(v_0 + k) \leftarrow -1$ for $0 \le k < n$. (Now all vertices are uncolored.) Then set $w \leftarrow v_0 + n$.

B2. [Done?] (At this point all vertices $\ge w$ have been colored, and so have the neighbors of all colored vertices.) Terminate the algorithm successfully if $w = v_0$. Otherwise set $w \leftarrow w - 1$, the next lower vertex node.

B3. [Color w if necessary.] If $\texttt{COLOR}(w) \ge 0$, return to B2. Otherwise set $\texttt{COLOR}(w) \leftarrow 0$, $\texttt{LINK}(w) \leftarrow \Lambda$, and $s \leftarrow w$.

B4. [Stack $\Rightarrow u$.] Set $u \leftarrow s$, $s \leftarrow \texttt{LINK}(s)$, $a \leftarrow \texttt{ARCS}(u)$. (We will examine all neighbors of the colored vertex u.)

B5. [Done with u?] If $a = \Lambda$, go to B8. Otherwise set $v \leftarrow \texttt{TIP}(a)$.

B6. [Process v.] If $\texttt{COLOR}(v) < 0$, set $\texttt{COLOR}(v) \leftarrow 1 - \texttt{COLOR}(u)$, $\texttt{LINK}(v) \leftarrow s$, and $s \leftarrow v$. Otherwise if $\texttt{COLOR}(v) = \texttt{COLOR}(u)$, terminate unsuccessfully.

B7. [Loop on a.] Set $a \leftarrow \texttt{NEXT}(a)$ and return to B5.

B8. [Stack nonempty?] If $s \ne \Lambda$, return to B4. Otherwise return to B2. ▌

This algorithm is a variant of a general graph traversal procedure called "depth-first search," which we will study in detail in Section 7.4.1. Its running time is $O(m + n)$ when there are m arcs and n vertices (see exercise 70); therefore it is well adapted to the common case of sparse graphs. With small changes we can make it output an odd-length cycle whenever it terminates unsuccessfully, thereby proving the impossibility of a 2-coloring (see exercise 72).

Examples of graphs. The Stanford GraphBase includes a library of more than three dozen generator routines, capable of producing a great variety of graphs and digraphs for use in experiments. We've already discussed *words*; now let's look at a few of the others, in order to get a feeling for some of the possibilities.

• *roget*(1022, 0, 0, 0) is a directed graph with 1022 vertices and 5075 arcs. The vertices represent the categories of words or concepts that P. M. Roget and J. L. Roget included in their famous 19th-century *Thesaurus* (London: Longmans, Green, 1879). The arcs are the cross references between categories, as found in that book. For example, typical arcs are `water` ⟶ `moisture`, `discovery` ⟶ `truth`, `preparation` ⟶ `learning`, `vulgarity` ⟶ `ugliness`, `wit` ⟶ `amusement`.

• *book*("`jean`", 80, 0, 1, 356, 0, 0, 0) is a graph with 80 vertices and 254 edges. The vertices represent the characters of Victor Hugo's *Les Misérables*; the edges connect characters who encounter each other in that novel. Typical edges are `Fantine` — `Javert`, `Cosette` — `Thénardier`.

• *bi_book*("`jean`", 80, 0, 1, 356, 0, 0, 0) is a bipartite graph with 80+356 vertices and 727 edges. The vertices represent characters or chapters in *Les Misérables*; the edges connect characters with the chapters in which they appear (for instance, `Napoleon` — `2.1.8`, `Marius` — `4.14.4`).

• *plane_miles*(128, 0, 0, 0, 1, 0, 0) is a planar graph with 129 vertices and 381 edges. The vertices represent 128 cities in the United States or Canada, plus a special vertex `INF` for a "point at infinity." The edges define the so-called *Delaunay triangulation* of those cities, based on latitude and longitude in a plane; this means that u — v if and only if the smallest circle that passes through u and v does not enclose any other vertex. Edges also run between `INF` and all vertices that lie on the convex hull of all city locations. Typical edges are **`Seattle, WA`** — **`Vancouver, BC`** — **`INF`**; **`Toronto, ON`** — **`Rochester, NY`**.

• *plane_lisa*(360, 250, 15, 0, 360, 0, 250, 0, 0, 2295000) is a planar graph that has 3027 vertices and 5967 edges. It is obtained by starting with a digitized image of Leonardo da Vinci's *Mona Lisa*, having 360 rows and 250 columns of pixels, then rounding the pixel intensities to 16 levels of gray from 0 (black) to 15 (white). The resulting 3027 rookwise connected regions of constant brightness are then considered to be neighbors when they share a pixel boundary. (See Fig. 4.)

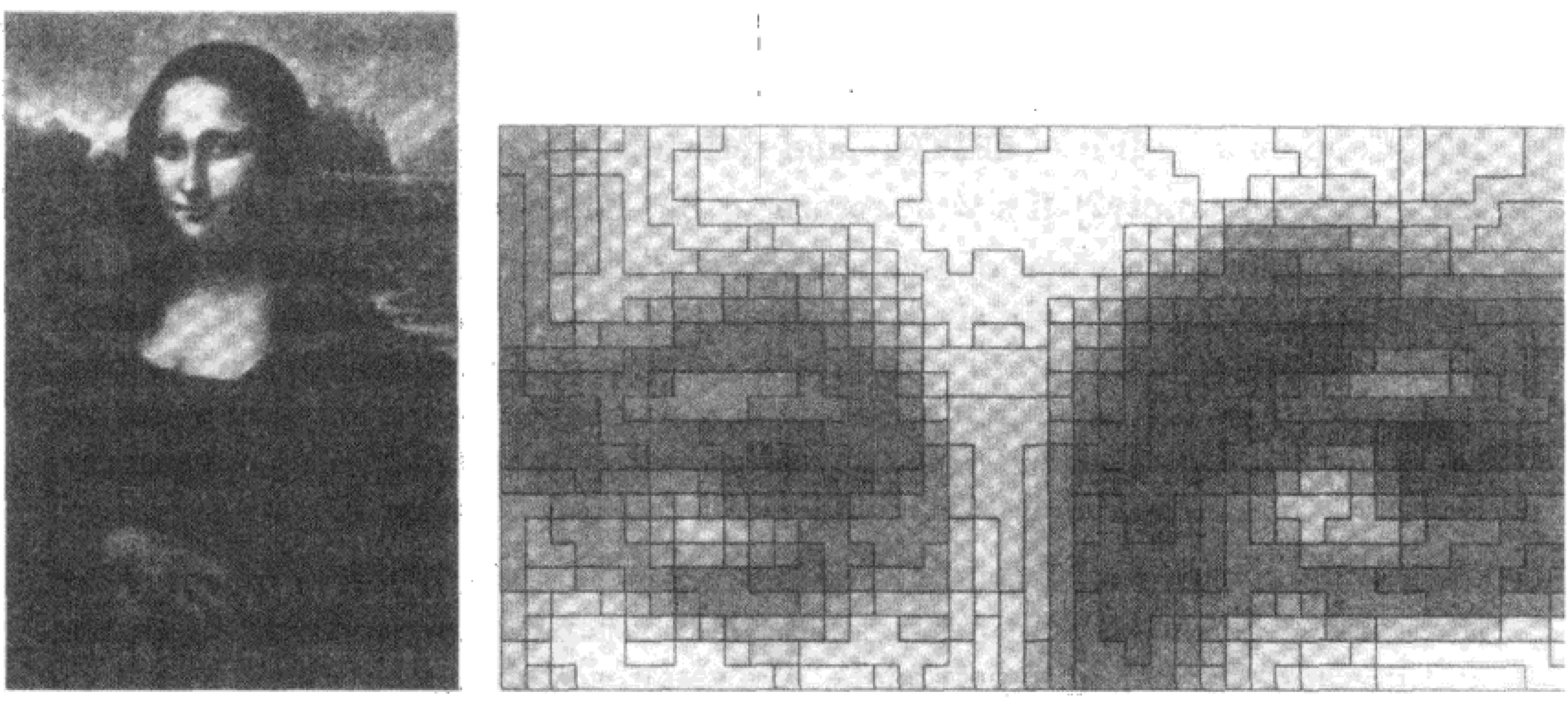


Fig. 4. A digital rendition of *Mona Lisa*, with a closeup detail (best viewed from afar).

• *bi_lisa*(360, 250, 0, 360, 0, 250, 8192, 0) is a bipartite graph with 360 + 250 = 610 vertices and 40923 edges. It's another takeoff on Leonardo's famous painting, this time linking rows and columns where the brightness level is at least 1/8. For example, the edge `r102` — `c113` occurs right in the middle of Lisa's "smile."

• *raman*(31, 23, 3, 1) is a graph with quite a different nature from the SGB graphs in previous examples. Instead of being linked to language, literature, or other outgrowths of human culture, it's a so-called "Ramanujan expander graph," based on strict mathematical principles. Each of its $(23^3 - 23)/2 = 6072$ vertices has degree 32; hence it has 97152 edges. The vertices correspond to equivalence classes of 2×2 matrices that are nonsingular modulo 23; a typical edge is **`(2,7;1,1)`** — **`(4,6;1,3)`**. Ramanujan graphs are important chiefly because they have unusually high girth and low diameter for their size and degree. This one has girth 4 and diameter 4.

• *raman*(5, 37, 4, 1), similarly, is a regular graph of degree 6 with 50616 vertices and 151848 edges. It has girth 10, diameter 10, and happens also to be bipartite.

• *random_graph*(1000, 5000, 0, 0, 0, 0, 0, 0, 0, s) is a graph with 1000 vertices, 5000 edges, and seed s. It "evolved" by starting with no edges, then by repeatedly choosing pseudorandom vertex numbers $0 \le u, v < 1000$ and adding the edge $u - v$, unless $u = v$ or that edge was already present. When $s = 0$, all vertices belong to a giant component of order 999, except for the isolated vertex 908.

• *random_graph*(1000, 5000, 0, 0, 1, 0, 0, 0, 0, 0) is a digraph with 1000 vertices and 5000 arcs, obtained via a similar sort of evolution. (In fact, each of its arcs happens to be part also of *random_graph*(1000, 5000, 0, 0, 0, 0, 0, 0, 0, 0).)

• *subsets*(5, 1, −10, 0, 0, 0, #1, 0) is a graph with $\binom{11}{5} = 462$ vertices, one for every five-element subset of $\{0, 1, \ldots, 10\}$. Two vertices are adjacent whenever the corresponding subsets are disjoint; thus, the graph is regular of degree 6, and it has 1386 edges. We can consider it to be a generalization of the Petersen graph, which has *subsets*(2, 1, −4, 0, 0, 0, #1, 0) as one of its SGB names.

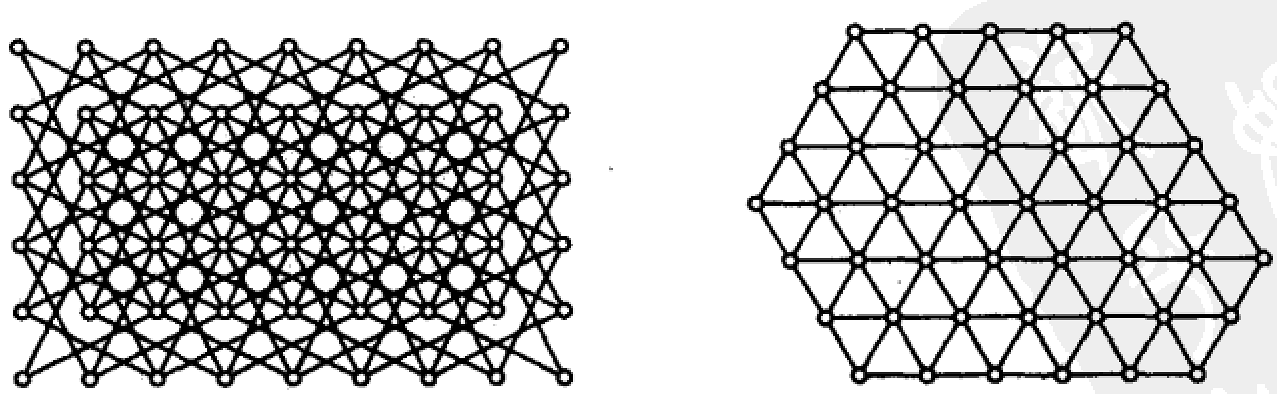
• *subsets*(5, 1, −10, 0, 0, 0, #10, 0) has the same 462 vertices, but now they are adjacent if the corresponding subsets have four elements in common. This graph is regular of degree 30, and it has 6930 edges.

• *parts*(30, 10, 30, 0) is another SGB graph with a mathematical basis. It has 3590 vertices, one for each partition of 30 into at most 10 parts. Two partitions are adjacent when one is obtained by subdividing a part of the other; this rule defines 31377 edges. The digraph *parts*(30, 10, 30, 1) is similar, but its 31377 arcs point from shorter to longer partitions (for example, 13+7+7+3 ⟶ 7+7+7+6+3).

• *simplex*(10, 10, 10, 10, 10, 0, 0) is a graph with 286 vertices and 1320 edges. Its vertices are the integer solutions to $x_1 + x_2 + x_3 + x_4 = 10$ with $x_i \ge 0$, namely the "compositions of 10 into four nonnegative parts"; they can also be regarded as barycentric coordinates for points inside a tetrahedron. The edges, such as 3,1,4,2 — 3,0,4,3, connect compositions that are as close together as possible.

• *board*(8, 8, 0, 0, 5, 0, 0) and *board*(8, 8, 0, 0, −2, 0, 0) are graphs on 64 vertices whose 168 or 280 edges correspond to the moves of a knight or bishop in chess.

And zillions of further examples are obtainable by varying the parameters to the SGB graph generators. For example, Fig. 5 shows two simple variants of *board* and *simplex*; the somewhat arcane rules of *board* are explained in exercise 75.



board(6, 9, 0, 0, 5, 0, 0)
(Knight moves on a 6 × 9 chessboard)

simplex(10, 8, 7, 6, 0, 0, 0)
(A truncated triangular grid)

Fig. 5. Samples of SGB graphs related to board games.

Graph algebra. We can also obtain new graphs by operating on the graphs that we already have. For example, if $G = (V, E)$ is any graph, its *complement* $\overline{G} = (V, \overline{E})$ is obtained by letting

$$u \text{---} v \text{ in } \overline{G} \qquad \Longleftrightarrow \qquad u \neq v \text{ and } u \not\text{---} v \text{ in } G. \tag{35}$$

Thus, non-edges become edges, and vice versa. Notice that $\overline{\overline{G}} = G$, and that $\overline{K_n}$ has no edges. The corresponding adjacency matrices A and $\overline{A}$ satisfy

$$A + \overline{A} = J - I; \tag{36}$$

here J is the matrix of all 1s, and I is the identity matrix, so J and $J - I$ are respectively the adjacency matrices of J_n and K_n when G has order n.

Furthermore, every graph $G = (V, E)$ leads to a *line graph* $L(G)$, whose vertices are the edges E; two edges are adjacent in $L(G)$ if they have a common vertex. Thus, for example, the line graph $L(K_n)$ has $\binom{n}{2}$ vertices, and it is regular of degree $2n - 4$ when $n \geq 2$ (see exercise 82). A graph is called *k-edge-colorable* when its line graph is k-colorable.

Given two graphs $G = (U, E)$ and $H = (V, F)$, their *union* $G \cup H$ is the graph $(U \cup V, E \cup F)$ obtained by combining the vertices and edges. For example, suppose G and H are the graphs of rook and bishop moves in chess; then $G \cup H$ is the graph of queen moves, and its official SGB name is

$$\mathit{gunion}(\mathit{board}(8, 8, 0, 0, -1, 0, 0), \mathit{board}(8, 8, 0, 0, -2, 0, 0), 0, 0). \tag{37}$$

In the special case where the vertex sets U and V are disjoint, the union $G \cup H$ doesn't require the vertices to be identified in any consistent way for cross-correlation; we get a diagram for $G \cup H$ by simply drawing a diagram of G next to a diagram of H. This special case is called the "juxtaposition" or *direct sum* of G and H, and we shall denote it by $G \oplus H$. For example, it's easy to see that

$$K_m \oplus K_n \cong \overline{K_{m,n}}, \tag{38}$$

and that every graph is the direct sum of its connected components.

Equation (38) is a special case of the general formula

$$K_{n_1} \oplus K_{n_2} \oplus \cdots \oplus K_{n_k} \cong \overline{K_{n_1, n_2, \ldots, n_k}}, \tag{39}$$

which holds for complete k-partite graphs whenever $k \geq 2$. But (39) fails when $k = 1$, because of a scandalous fact: The standard graph-theoretic notation for complete graphs is inconsistent! Indeed, $K_{m,n}$ denotes a complete 2-partite graph, but K_n does *not* denote a complete 1-partite graph. Somehow graph theorists have been able to live with this anomaly for decades without going berserk.

Another important way to combine disjoint graphs G and H is to form their *join*, $G \text{---} H$, which consists of $G \oplus H$ together with all edges $u \text{---} v$ for $u \in U$ and $v \in V$. [See A. A. Zykov, *Mat. Sbornik* **24** (1949), 163–188, §I.3.] And if G and H are disjoint *digraphs*, their *directed join* $G \longrightarrow H$ is similar, but it supplements $G \oplus H$ by adding only the one-way arcs $u \longrightarrow v$ from U to V.

The direct sum of two matrices A and B is obtained by placing B diagonally below and to the right of A:

$$A \oplus B = \begin{pmatrix} A & O \\ O & B \end{pmatrix}, \qquad (40)$$

where each O in this example is a matrix of all zeros, with the proper number of rows and columns to make everything line up correctly. Our notation $G \oplus H$ for the direct sum of graphs is easy to remember because the adjacency matrix for $G \oplus H$ is precisely the direct sum of the respective adjacency matrices A and B for G and H. Similarly, the adjacency matrices for G — H, $G \longrightarrow H$, and $G \longleftarrow H$ are

$$A \text{—} B = \begin{pmatrix} A & J \\ J & B \end{pmatrix}, \qquad A \longrightarrow B = \begin{pmatrix} A & J \\ O & B \end{pmatrix}, \qquad A \longleftarrow B = \begin{pmatrix} A & O \\ J & B \end{pmatrix}, \quad (41)$$

respectively, where J is an all-1s matrix as in (36). These operations are associative, and related by complementation:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C, \qquad A \text{—} (B \text{—} C) = (A \text{—} B) \text{—} C; \qquad (42)$$

$$A \longrightarrow (B \longrightarrow C) = (A \longrightarrow B) \longrightarrow C, \qquad A \longleftarrow (B \longleftarrow C) = (A \longleftarrow B) \longleftarrow C; \qquad (43)$$

$$\overline{A \oplus B} = \overline{A} \text{—} \overline{B}, \qquad \overline{A \text{—} B} = \overline{A} \oplus \overline{B}; \qquad (44)$$

$$\overline{A \longrightarrow B} = \overline{A} \longleftarrow \overline{B}, \qquad \overline{A \longleftarrow B} = \overline{A} \longrightarrow \overline{B}; \qquad (45)$$

$$(A \oplus B) + (A \text{—} B) = (A \longrightarrow B) + (A \longleftarrow B). \qquad (46)$$

Notice that, by combining (39) with (42) and (44), we have

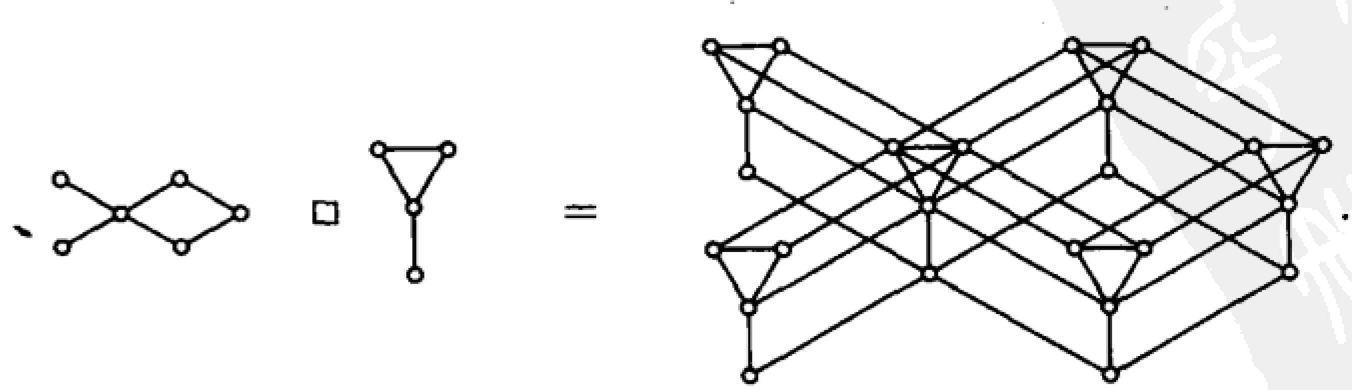
$$K_{n_1,n_2,\ldots,n_k} = \overline{K_{n_1}} \text{—} \overline{K_{n_2}} \text{—} \cdots \text{—} \overline{K_{n_k}} \qquad (47)$$

when $k \ge 2$. Also

$$K_n = K_1 \text{—} K_1 \text{—} \cdots \text{—} K_1 \quad \text{and} \quad K_n^{\rightarrow} = K_1 \longrightarrow K_1 \longrightarrow \cdots \longrightarrow K_1, \qquad (48)$$

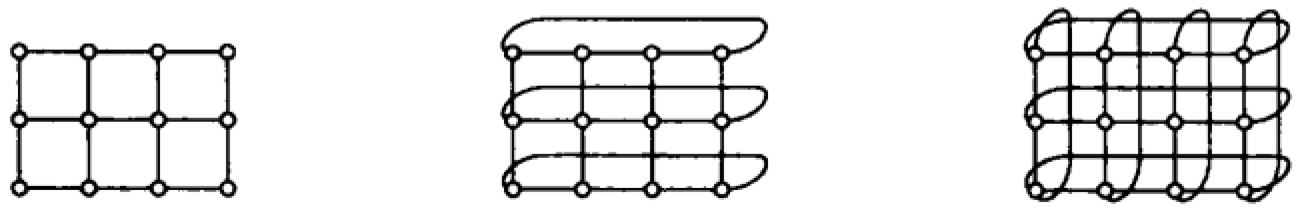
with n copies of K_1, showing that $K_n = K_{1,1,\ldots,1}$ is a complete n-partite graph.

Direct sums and joins are analogous to addition, because we have $\overline{K_m} \oplus \overline{K_n} = \overline{K_{m+n}}$ and K_m — $K_n = K_{m+n}$. We can also combine graphs with algebraic operations that are analogous to multiplication. For example, the *Cartesian product* operation forms a graph $G \,\square\, H$ of order mn from a graph $G = (U, E)$ of order m and a graph $H = (V, F)$ of order n. The vertices of $G \,\square\, H$ are ordered pairs (u, v), where $u \in U$ and $v \in V$; the edges are (u, v) — (u', v) when u — u' in G, together with (u, v) — (u, v') when v — v' in H. In other words, $G \,\square\, H$ is formed by replacing each vertex of G by a copy of H, and replacing each edge of G by edges between corresponding vertices of the appropriate copies:



(49)

As usual, the simplest special cases of this general construction turn out to be especially important in practice. When both G and H are paths or cycles, we get "graph-paper graphs," namely the $m \times n$ *grid* $P_m \,\square\, P_n$, the $m \times n$ *cylinder* $P_m \,\square\, C_n$, and the $m \times n$ *torus* $C_m \,\square\, C_n$, illustrated here for $m = 3$ and $n = 4$:



$P_3 \,\square\, P_4$ (3 × 4 grid) $\qquad$ $P_3 \,\square\, C_4$ (3 × 4 cylinder) $\qquad$ $C_3 \,\square\, C_4$ (3 × 4 torus) (50)

Four other noteworthy ways to define products of graphs have also proved to be useful. In each case the vertices of the product graph are ordered pairs (u, v).

- The *direct product* $G \otimes H$, also called the "conjunction" of G and H, or their "categorical product," has $(u, v) — (u', v')$ when $u — u'$ in G and $v — v'$ in H.
- The *strong product* $G \boxtimes H$ combines the edges of $G \,\square\, H$ with those of $G \otimes H$.
- The *odd product* $G \,\triangle\, H$ has $(u, v) — (u', v')$ when we have either $u — u'$ in G or $v — v'$ in H, but not both.
- The *lexicographic product* $G \circ H$, also called the "composition" of G and H, has $(u, v) — (u', v')$ when $u — u'$ in G, and $(u, v) — (u, v')$ when $v — v'$ in H.

All five of these operations extend naturally to products of $k \geq 2$ graphs $G_1 = (V_1, E_1)$, ..., $G_k = (V_k, E_k)$, whose vertices are the ordered k-tuples $(v_1, \ldots, v_k)$ with $v_j \in V_j$ for $1 \leq j \leq k$. For example, when $k = 3$, the Cartesian products $G_1 \,\square\, (G_2 \,\square\, G_3)$ and $(G_1 \,\square\, G_2) \,\square\, G_3$ are isomorphic, if we consider the compound vertices $(v_1, (v_2, v_3))$ and $((v_1, v_2), v_3)$ to be the same as (v_1, v_2, v_3). Therefore we can write this Cartesian product without parentheses, as $G_1 \,\square\, G_2 \,\square\, G_3$. The most important example of a Cartesian product with k factors is the k-cube,

$$K_2 \,\square\, K_2 \,\square\, \cdots \,\square\, K_2; \tag{51}$$

its 2^k vertices $(v_1, \ldots, v_k)$ are adjacent when their Hamming distance is 1.

In general, suppose $v = (v_1, \ldots, v_k)$ and $v' = (v'_1, \ldots, v'_k)$ are k-tuples of vertices, where we have $v_j — v'_j$ in G_j for exactly a of the subscripts j, and $v_j = v'_j$ for exactly b of the subscripts. Then we have:

- $v — v'$ in $G_1 \,\square\, \cdots \,\square\, G_k$ if and only if $a = 1$ and $b = k - 1$;
- $v — v'$ in $G_1 \otimes \cdots \otimes G_k$ if and only if $a = k$ and $b = 0$;
- $v — v'$ in $G_1 \boxtimes \cdots \boxtimes G_k$ if and only if $a + b = k$ and $a > 0$;
- $v — v'$ in $G_1 \,\triangle\, \cdots \,\triangle\, G_k$ if and only if a is odd.

The lexicographic product is somewhat different, because it isn't commutative; in $G_1 \circ \cdots \circ G_k$ we have $v — v'$ for $v \neq v'$ if and only if $v_j — v'_j$, where j is the minimum subscript with $v_j \neq v'_j$.

Exercises 91–102 explore some of the basic properties of graph products. See also the book *Product Graphs* by Wilfried Imrich and Sandi Klavžar (2000), which contains a comprehensive introduction to the general theory, including algorithms for factorization of a given graph into "prime" subgraphs.

***Graphical degree sequences.** A sequence $d_1 d_2 \dots d_n$ of nonnegative integers is called *graphical* if there's at least one graph on vertices $\{1, 2, \dots, n\}$ such that vertex k has degree d_k. We can assume that $d_1 \geq d_2 \geq \cdots \geq d_n$. Clearly $d_1 < n$ in any such graph; and the sum $m = d_1 + d_2 + \cdots + d_n$ of any graphical sequence is always even, because it is twice the number of edges. Furthermore, it's easy to see that the sequence 3311 is not graphical; therefore graphical sequences must also satisfy additional conditions. What are they?

A simple way to decide if a given sequence $d_1 d_2 \dots d_n$ is graphical, and to construct such a graph if one exists, was discovered by V. Havel [*Časopis pro Pěstování Matematiky* **80** (1955), 477–479]. We begin with an empty tableau, having d_k cells in row k; these cells represent "slots" into which we'll place the neighbors of vertex k in the constructed graph. Let c_j be the number of cells in column j; thus $c_1 \geq c_2 \geq \cdots$, and when $1 \leq k \leq n$ we have $c_j \geq k$ if and only if $d_k \geq j$. For example, suppose $n = 8$ and $d_1 \dots d_8 = 55544322$; then

1					
2					
3					
4					
5					
6					
7					
8					

(52)

is the initial tableau, and we have $c_1 \dots c_5 = 88653$. Havel's idea is to pair up vertex n with d_n of the highest-degree vertices. In this case, for example, we create the two edges $8 — 3$ and $8 — 2$, and the tableau takes the following form:

1					
2					8
3					8
4					
5					
6					
7					
8	2	3			

. (53)

(We don't want $8 — 1$, because the empty slots should continue to form a tableau shape; the cells of each column must be filled from the bottom up.) Next we set $n \leftarrow 7$ and create two further edges, $7 — 1$ and $7 — 5$. And then come three more, $6 — 4$, $6 — 3$, $6 — 2$, making the tableau almost half full:

1					7
2				6	8
3				6	8
4				6	
5				7	
6	2	3	4		
7	5	1			
8	2	3			

. (54)

We've reduced the problem to finding a graph with degree sequence $d_1 \ldots d_5 = 43333$; at this point we also have $c_1 \ldots c_4 = 5551$. The reader is encouraged to fill in the remaining blanks, before looking at the answer in exercise 103.

Algorithm H (*Graph generator for specified degrees*). Given $d_1 \geq \cdots \geq d_n \geq d_{n+1} = 0$, this algorithm creates edges between the vertices $\{1, \ldots, n\}$ in such a way that exactly d_k edges touch vertex k, for $1 \leq k \leq n$, unless the sequence $d_1 \ldots d_n$ isn't graphical. An array $c_1 \ldots c_{d_1}$ is used for auxiliary storage.

H1. [Set the c's.] Start with $k \leftarrow d_1$ and $j \leftarrow 0$. Then while $k > 0$ do the following operations: Set $j \leftarrow j + 1$; while $k > d_{j+1}$, set $c_k \leftarrow j$ and $k \leftarrow k - 1$.

H2. [Find n.] Set $n \leftarrow c_1$. Terminate successfully if $n = 0$; terminate unsuccessfully if $d_1 \geq n > 0$.

H3. [Loop on j.] Set $i \leftarrow 1$, $t \leftarrow d_1$, and $r \leftarrow c_t$. Do step H4 for $j = d_n, d_n - 1, \ldots, 1$; then return to H2.

H4. [Generate a new edge.] Set $c_j \leftarrow c_j - 1$ and $k \leftarrow c_t$. Create the edge $k — n$, and set $d_k \leftarrow d_k - 1$, $c_t \leftarrow k - 1$. If $k = i$, set $i \leftarrow r + 1$, $t \leftarrow d_i$, and $r \leftarrow c_t$. (See exercise 104.) ▌

When Algorithm H succeeds, it certainly has constructed a graph with the desired degrees. But when it fails, how can we be sure that its mission was impossible? The key fact is based on an important concept called "majorization": If $d_1 \ldots d_n$ and $d'_1 \ldots d'_n$ are two partitions of the same integer (that is, if $d_1 \geq \cdots \geq d_n$ and $d'_1 \geq \cdots \geq d'_n$ and $d_1 + \cdots + d_n = d'_1 + \cdots + d'_n$), we say that $d_1 \ldots d_n$ *majorizes* $d'_1 \ldots d'_n$ if $d_1 + \cdots + d_k \geq d'_1 + \cdots + d'_k$ for $1 \leq k \leq n$.

Lemma M. *If $d_1 \ldots d_n$ is graphical and $d_1 \ldots d_n$ majorizes $d'_1 \ldots d'_n$, then $d'_1 \ldots d'_n$ is also graphical.*

Proof. It is sufficient to prove the claim when $d_1 \ldots d_n$ and $d'_1 \ldots d'_n$ differ in only two places,

$$d'_k = d_k - [k=i] + [k=j] \qquad \text{where } i < j, \tag{55}$$

because any sequence majorized by $d_1 \ldots d_n$ can be obtained by repeatedly performing mini-majorizations such as this. (Exercise 7.2.1.4–55 discusses majorization in detail.)

Condition (55) implies that $d_i > d'_i \geq d'_{i+1} \geq d'_j > d_j$. So any graph with degree sequence $d_1 \ldots d_n$ contains a vertex v such that $v — i$ and $v \not— j$. Deleting the edge $v — i$ and adding the edge $v — j$ yields a graph with degree sequence $d'_1 \ldots d'_n$, as desired. ▌

Corollary H. *Algorithm H succeeds whenever $d_1 \ldots d_n$ is graphical.*

Proof. We may assume that $n > 1$. Suppose G is any graph on $\{1, \ldots, n\}$ with degree sequence $d_1 \ldots d_n$, and let G' be the subgraph induced by $\{1, \ldots, n-1\}$; in other words, obtain G' by removing vertex n and the d_n edges that it touches. The degree sequence $d'_1 \ldots d'_{n-1}$ of G' is obtained from $d_1 \ldots d_{n-1}$ by reducing some d_n of the entries by 1 and sorting them into nonincreasing order. By

definition, $d'_1 \ldots d'_{n-1}$ is graphical. The new degree sequence $d''_1 \ldots d''_{n-1}$ produced by the strategy of steps H3 and H4 is designed to be majorized by every such $d'_1 \ldots d'_{n-1}$, because it reduces the largest possible d_n entries by 1. Thus the new $d''_1 \ldots d''_{n-1}$ is graphical. Algorithm H, which sets $d_1 \ldots d_{n-1} \leftarrow d''_1 \ldots d''_{n-1}$, will therefore succeed by induction on n. ▮

The running time of Algorithm H is roughly proportional to the number of edges generated, which can be of order n^2. Exercise 105 presents a faster method, which decides in $O(n)$ steps whether or not a given sequence $d_1 \ldots d_n$ is graphical (without constructing any graph).

Beyond graphs. When the vertices and/or arcs of a graph or digraph are decorated with additional data, we call it a *network*. For example, every vertex of *words*(5757, 0, 0, 0) has an associated rank, which corresponds to the popularity of the corresponding five-letter word. Every vertex of *plane_lisa*(360, 250, 15, 0, 360, 0, 250, 0, 0, 2295000) has an associated pixel density, between 0 and 15. Every arc of *board*(8, 8, 0, 0, −2, 0, 0) has an associated length, which reflects the distance of a piece's motion on the board: A bishop's move from corner to corner has length 7. The Stanford GraphBase includes several further generators that were not mentioned above, because they are primarily used to generate interesting networks, rather than to generate graphs with interesting structure:

• *miles*(128, 0, 0, 0, 0, 127, 0) is a network with 128 vertices, corresponding to the same North American cities as the graph *plane_miles* described earlier. But *miles*, unlike *plane_miles*, is a complete graph with $\binom{128}{2}$ edges. Every edge has an integer length, which represents the distance that a car or truck would have needed to travel in 1949 when going from one given city to another. For example, '`Vancouver, BC`' is 3496 miles from '`West Palm Beach, FL`' in the *miles* network.

• *econ*(81, 0, 0, 0) is a network with 81 vertices and 4902 arcs. Its vertices represent sectors of the United States economy, and its arcs represent the flow of money from one sector to another during the year 1985, measured in millions of dollars. For example, the flow value from `Apparel` to `Household furniture` is 44, meaning that the furniture industry paid $44,000,000 to the apparel industry in that year. The sum of flows coming into each vertex is equal to the sum of flows going out. An arc appears only when the flow is nonzero. A special vertex called `Users` receives the flows that represent total demand for a product; a few of these end-user flows are negative, because of the way imported goods are treated by government economists.

• *games*(120, 0, 0, 0, 0, 0, 128, 0) is a network with 120 vertices and 1276 arcs. Its vertices represent football teams at American colleges and universities. Arcs run between teams that played each other during the exciting 1990 season, and they are labeled with the number of points scored. For example, the arc `Stanford` $\longrightarrow$ `California` has value 27, and the arc `California` $\longrightarrow$ `Stanford` has value 25, because the Stanford Cardinal defeated the U. C. Berkeley Golden Bears by a score of 27–25 on 17 November 1990.

• *risc*(16) is a network of an entirely different kind. It has 3240 vertices and 7878 arcs, which define a *directed acyclic graph* or "dag" — namely, a digraph

that contains no oriented cycles. The vertices represent gates that have Boolean values; an arc such as `Z45` $\longrightarrow$ `R0:7~` means that the value of gate `Z45` is an input to gate `R0:7~`. Each gate has a type code (AND, OR, XOR, NOT, latch, or external input); each arc has a length, denoting an amount of delay. The network contains the complete logic for a miniature RISC chip that is able to obey simple commands governing sixteen registers, each 16 bits wide.

Complete details about all the SGB generators can be found in the author's book *The Stanford GraphBase* (New York: ACM Press, 1993), together with dozens of short example programs that explain how to manipulate the graphs and networks that the generators produce. For example, a program called LADDERS shows how to find a shortest path between one five-letter word and another. A program called TAKE_RISC demonstrates how to put a nanocomputer through its paces by simulating the actions of a network built from the gates of *risc*(16).

Hypergraphs. Graphs and networks can be utterly fascinating, but they aren't the end of the story by any means. Lots of important combinatorial algorithms are designed to work with *hypergraphs*, which are more general than graphs because their edges are allowed to be *arbitrary* subsets of the vertices.

For example, we might have seven vertices, identified by nonzero binary strings $v = a_1a_2a_3$, together with seven edges, identified by bracketed nonzero binary strings $e = [b_1b_2b_3]$, with $v \in e$ if and only if $(a_1b_1+a_2b_2+a_3b_3) \bmod 2 = 0$. Each of these edges contains exactly three vertices:

$$\begin{gathered}[001] = \{010, 100, 110\}; \quad [010] = \{001, 100, 101\}; \quad [011] = \{011, 100, 111\}; \\ [100] = \{001, 010, 011\}; \quad [101] = \{010, 101, 111\}; \\ [110] = \{001, 110, 111\}; \quad [111] = \{011, 101, 110\}. \end{gathered} \tag{56}$$

And by symmetry, each vertex belongs to exactly three edges. (Edges that contain three or more vertices are sometimes called "hyperedges," to distinguish them from the edges of an ordinary graph. But it's OK to call them just "edges.")

A hypergraph is said to be r-*uniform* if every edge contains exactly r vertices. Thus (56) is a 3-uniform hypergraph, and a 2-uniform hypergraph is an ordinary graph. The complete r-uniform hypergraph $K_n^{(r)}$ has n vertices and $\binom{n}{r}$ edges.

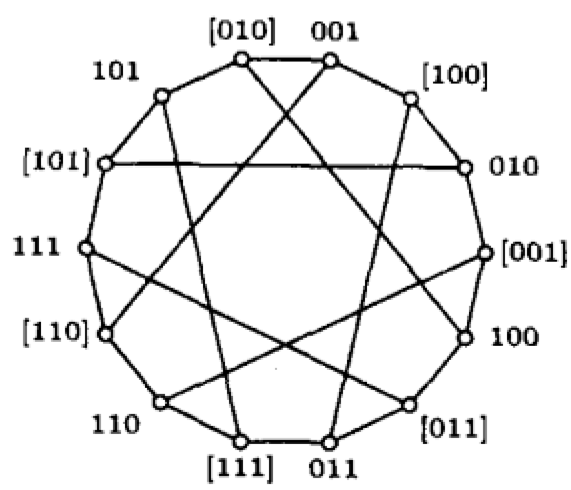
Most of the basic concepts of graph theory can be extended to hypergraphs in a natural way. For example, if $H = (V, E)$ is a hypergraph and if $U \subseteq V$, the subhypergraph $H \mid U$ induced by U has the edges $\{e \mid e \in E \text{ and } e \subseteq U\}$. The complement $\overline{H}$ of an r-uniform hypergraph has the edges of $K_n^{(r)}$ that aren't edges of H. A k-coloring of a hypergraph is an assignment of colors to the vertices so that no edge is monochromatic. And so on.

Hypergraphs go by many other names, because the same properties can be formulated in many different ways. For example, every hypergraph $H = (V, E)$ is essentially a *family of sets*, because each edge is a subset of V. A 3-uniform hypergraph is also called a *triple system*. A hypergraph is also equivalent to a matrix B of 0s and 1s, with one row for each vertex v and one column for each edge e; row v and column e of this matrix contains the value $b_{ve} = [v \in e]$.

Matrix B is called the *incidence matrix* of H, and we say that "v is incident with e" when $v \in e$. Furthermore, a hypergraph is equivalent to a *bipartite graph*, with vertex set $V \cup E$ and with the edge v — e whenever v is incident with e. The hypergraph is said to be *connected* if and only if the corresponding bipartite graph is connected. A *cycle* of length k in a hypergraph is defined to be a cycle of length $2k$ in the corresponding bipartite graph.

For example, the hypergraph (56) can be defined by an equivalent incidence matrix or an equivalent bipartite graph as follows:

	[001]	[010]	[011]	[100]	[101]	[110]	[111]
001	0	1	0	1	0	1	0
010	1	0	0	1	1	0	0
011	0	0	1	1	0	0	1
100	1	1	1	0	0	0	0
101	0	1	0	0	1	0	1
110	1	0	0	0	0	1	1
111	0	0	1	0	1	1	0



(57)

It contains 28 cycles of length 3, such as

$$[101] — 101 — [010] — 001 — [100] — 010 — [101]. \tag{58}$$

The *dual* H^T of a hypergraph H is obtained by interchanging the roles of vertices and edges, but retaining the incidence relation. In other words, it corresponds to transposing the incidence matrix. Notice, for example, that the dual of an r-regular graph is an r-uniform hypergraph.

Incidence matrices and bipartite graphs might correspond to hypergraphs in which some edges occur more than once, because distinct columns of the matrix might be equal. When a hypergraph $H = (V, E)$ does not have any repeated edges, it corresponds also to yet another combinatorial object, namely a *Boolean function*. For if, say, the vertex set V is $\{1, 2, \ldots, n\}$, the function

$$h(x_1, x_2, \ldots, x_n) \;=\; \bigl[\{j \mid x_j = 1\} \in E\bigr] \tag{59}$$

characterizes the edges of H. For example, the Boolean formula

$$\begin{aligned}(x_1 \oplus x_2 \oplus x_4) \wedge (x_2 \oplus x_3 \oplus x_5) \wedge (x_3 \oplus x_4 \oplus x_6) \wedge (x_4 \oplus x_5 \oplus x_7)\\ \wedge (x_5 \oplus x_6 \oplus x_1) \wedge (x_6 \oplus x_7 \oplus x_2) \wedge (x_7 \oplus x_1 \oplus x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)\end{aligned} \tag{60}$$

is another way to describe the hypergraph of (56) and (57).

The fact that combinatorial objects can be viewed in so many ways can be mind-boggling. But it's also extremely helpful, because it suggests different ways to solve equivalent problems. When we look at a problem from different perspectives, our brains naturally think of different ways to attack it. Sometimes we get the best insights by thinking about how to manipulate rows and columns in a matrix. Sometimes we make progress by imagining vertices and paths, or by visualizing clusters of points in space. Sometimes Boolean algebra is just the thing. If we're stuck in one domain, another might come to our rescue.

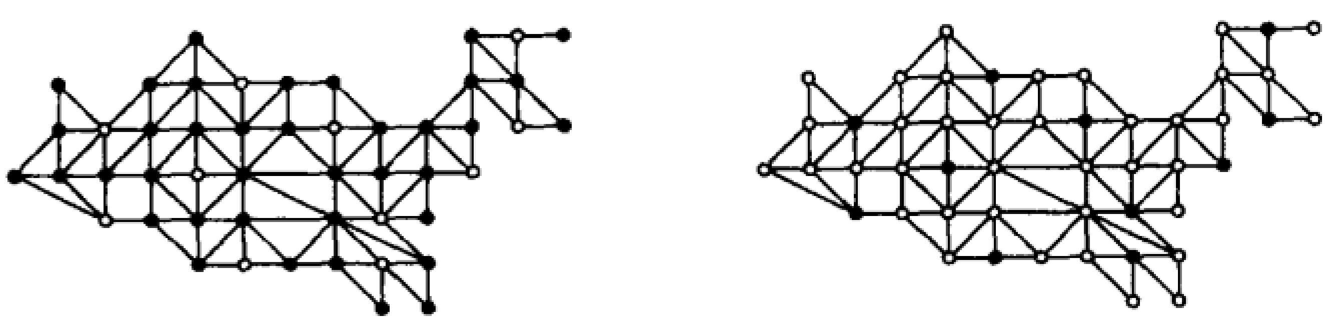
Covering and independence. If $H = (V, E)$ is a graph or hypergraph, a set U of vertices is said to *cover* H if every edge contains at least one member of U. A set W of vertices is said to be *independent* (or "stable") in H if no edge is completely contained in W.

From the standpoint of the incidence matrix, a covering is a set of rows whose sum is nonzero in every column. And in the special case that H is a graph, every column of the matrix contains just two 1s; hence an independent set in a graph corresponds to a set of rows that are mutually orthogonal — that is, a set for which the dot product of any two different rows is zero.

These concepts are opposite sides of the same coin. If U covers H, then $W = V \setminus U$ is independent in H; conversely, if W is independent in H, then $U = V \setminus W$ covers H. Both statements are equivalent to saying that the induced hypergraph $H \mid W$ has no edges.

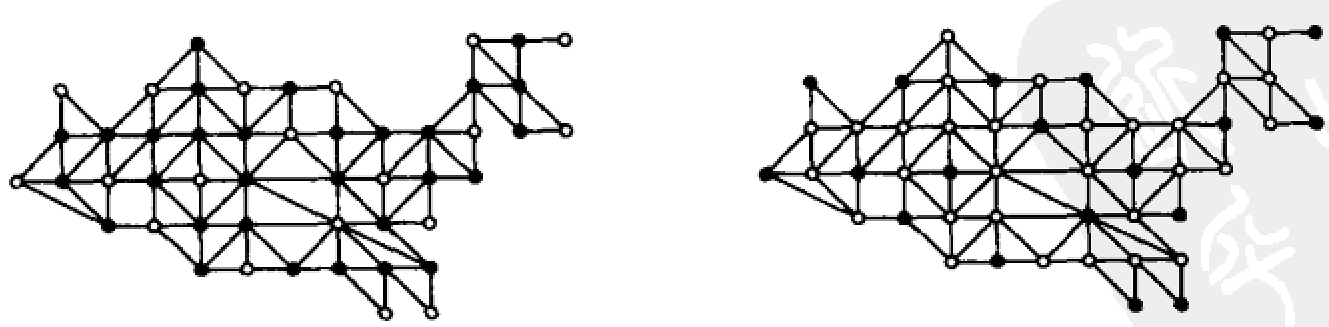
This dual relationship between covering and independence, which was perhaps first noted by Claude Berge [*Proc. National Acad. Sci.* **43** (1957), 842–844], is somewhat paradoxical. Although it's logically obvious and easy to verify, it's also intuitively surprising. When we look at a graph and try to find a large independent set, we tend to have rather different thoughts from when we look at the same graph and try to find a small vertex cover; yet both goals are the same.

A covering set U is *minimal* if $U \setminus u$ fails to be a cover for all $u \in U$. Similarly, an independent set W is *maximal* if $W \cup w$ fails to be independent for all $w \notin W$. Here, for example, is a minimal cover of the 49-vertex graph of the contiguous United States, (17), and the corresponding maximal independent set:



Minimal vertex cover, with 38 vertices — Maximal independent set, with 11 vertices (61)

A covering is called *minimum* if it has the smallest possible size, and an independent set is called *maximum* if it has the largest possible size. For example, with graph (17) we can do much better than (61):



Minimum vertex cover, with 30 vertices — Maximum independent set, with 19 vertices (62)

Notice the subtle distinction between "minimal" and "minimum" here: In general (but in contrast to most dictionaries of English), people who work with combinatorial algorithms use '-al' words like "minimal" or "optimal" to refer

to combinatorial configurations that are *locally* best, in the sense that small changes don't improve them. The corresponding '-um' words, "minimum" or "optimum," are reserved for configurations that are *globally* best, considered over all possibilities. It's easy to find solutions to any optimization problem that are merely optimal, in the weak local sense, by climbing repeatedly until reaching the top of a hill. But it's usually much harder to find solutions that are truly optimum. For example, we'll see in Section 7.9 that the problem of finding a maximum independent set in a given graph belongs to a class of difficult problems that are called *NP-complete.*

Even when a problem is NP-complete, we needn't despair. We'll discuss techniques for finding minimum covers in several parts of this chapter, and those methods work fine on smallish problems; the optimum solution in (62) was found in less than a second, after examining only a tiny fraction of the 2^{49} possibilities. Furthermore, special cases of NP-complete problems often turn out to be simpler than the general case. In Section 7.5.1 we'll see that a minimum vertex cover can be discovered quickly in any bipartite graph, or in any hypergraph that is the dual of a graph. And in Section 7.5.5 we'll study efficient ways to discover a maximum *matching*, which is a maximum independent set in the line graph of a given graph.

The problem of maximizing the size of an independent set occurs sufficiently often that it has acquired a special notation: If H is any hypergraph, the number

$$\alpha(H) \;=\; \max\bigl\{|W| \bigm| W \text{ is an independent set of vertices in } H\bigr\} \qquad (63)$$

is called the *independence number* (or the stability number) of H. Similarly,

$$\chi(H) \;=\; \min\{k \mid H \text{ is } k\text{-colorable}\} \qquad (64)$$

is called the *chromatic number* of H. Notice that $\chi(H)$ is the size of a minimum covering of H by independent sets, because the vertices that receive any particular color must be independent according to our definitions.

These definitions of $\alpha(H)$ and $\chi(H)$ apply in particular to the case when H is an ordinary graph, but of course we usually write $\alpha(G)$ and $\chi(G)$ in such situations. Graphs have another important number called their *clique number*,

$$\omega(G) \;=\; \max\bigl\{|X| \bigm| X \text{ is a clique in } G\bigr\}, \qquad (65)$$

where a "clique" is a set of mutually adjacent vertices. Clearly

$$\omega(G) \;=\; \alpha(\overline{G}), \qquad (66)$$

because a clique in G is an independent set in the complementary graph. Similarly we can see that $\chi(\overline{G})$ is the minimum size of a "clique cover," which is a set of cliques that exactly covers all of the vertices.

Several instances of "exact cover problems" were mentioned earlier in this section, without an explanation of exactly what such a problem really signifies. Finally we're ready for the definition: Given the incidence matrix of a hypergraph H, an *exact cover* of H is a set of rows whose sum is $(1\,1\,\ldots\,1)$. In other words, an exact cover is a set of vertices that touches each hyperedge exactly once; an ordinary cover is only required to touch each hyperedge *at least* once.

EXERCISES

1. [*25*] Suppose $n = 4m - 1$. Construct arrangements of Langford pairs for the numbers $\{1, 1, \ldots, n, n\}$, with the property that we also obtain a solution for $n = 4m$ by changing the first '$2m-1$' to '$4m$' and appending '$2m-1$ $4m$' at the right. *Hint:* Put the $m - 1$ even numbers $4m-4$, $4m-6$, $\ldots$, $2m$ at the left.

2. [*18*] For which n can $\{0, 0, 1, 1, \ldots, n-1, n-1\}$ be arranged as Langford pairs?

3. [*22*] Suppose we arrange the numbers $\{0, 0, 1, 1, \ldots, n-1, n-1\}$ in a *circle*, instead of a straight line, with distance k between the two k's. Do we get solutions that are essentially distinct from those of exercise 2?

4. [*M20*] (T. Skolem, 1957.) Show that the Fibonacci string $S_\infty = babbababbabba\ldots$ of exercise 1.2.8–36 leads directly to an infinite sequence 0012132453674... of Langford pairs for the set of *all* nonnegative integers, if we simply replace the a's and b's independently by 0, 1, 2, etc., from left to right.

▸ **5.** [*HM22*] If a permutation of $\{1, 1, 2, 2, \ldots, n, n\}$ is chosen at random, what is the probability that the two k's are exactly k positions apart, given k? Use this formula to guess the size of the Langford numbers L_n in (1).

▸ **6.** [*M28*] (M. Godfrey, 2002.) Let $f(x_1, \ldots, x_{2n}) = \prod_{k=1}^{n} \bigl(x_k x_{n+k} \sum_{j=1}^{2n-k-1} x_j x_{j+k+1}\bigr)$.

a) Prove that $\sum_{x_1,\ldots,x_{2n}\in\{-1,+1\}} f(x_1, \ldots, x_{2n}) = 2^{2n+1} L_n$.

b) Explain how to evaluate this sum in $O(4^n n)$ steps. How many bits of precision are needed for the arithmetic?

c) Gain a factor of eight by exploiting the identities

$$f(x_1, \ldots, x_{2n}) = f(-x_1, \ldots, -x_{2n}) = f(x_{2n}, \ldots, x_1) = f(x_1, -x_2, \ldots, x_{2n-1}, -x_{2n}).$$

7. [*M22*] Prove that every Langford pairing of $\{1, 1, \ldots, 16, 16\}$ must have seven uncompleted pairs at some point, when read from left to right.

8. [*23*] The simplest Langford sequence is not only well-balanced; it's *planar*, in the sense that its pairs can be connected up without crossing lines as in (2):

2 3 1 2 1 3.

Find all of the planar Langford pairings for which $n \le 8$.

9. [*24*] (*Langford triples.*) In how many ways can $\{1, 1, 1, 2, 2, 2, \ldots, 9, 9, 9\}$ be arranged in a row so that consecutive k's are k apart, for $1 \le k \le 9$?

10. [*M20*] Explain how to construct a *magic square* directly from Fig. 1. (Convert each card into a number between 1 and 16, in such a way that the rows, columns, and main diagonals all sum to 34.)

11. [*20*] Extend (5) to a "Hebraic-Græco-Latin" square by appending one of the letters {א, ב, ג, ד} to the two-letter string in each compartment. No letter pair (Latin, Greek), (Latin, Hebrew), or (Greek, Hebrew) should appear in more than one place.

▸ **12.** [*M21*] (L. Euler.) Let $L_{ij} = (i+j) \bmod n$ for $0 \le i, j < n$ be the addition table for integers mod n. Prove that a latin square orthogonal to L exists if and only if n is odd.

13. [*M25*] A 10×10 square can be divided into four quarters of size 5×5. A 10×10 latin square formed from the digits $\{0, 1, \ldots, 9\}$ has k "intruders" if its upper left quarter has exactly k elements ≥ 5. (See exercise 14(e) for an example with $k = 3$.) Prove that the square has no orthogonal mate unless there are at least three intruders.

14. [*29*] Find all orthogonal mates of the following latin squares:

(a)	(b)	(c)	(d)	(e)
3145926870	2718459036	0572164938	1680397425	7823456019
2819763504	0287135649	6051298473	8346512097	8234067195
9452307168	7524093168	4867039215	9805761342	2340178956
6208451793	1435962780	1439807652	2754689130	3401289567
8364095217	6390718425	8324756091	0538976214	4012395678
5981274036	4069271853	7203941586	4963820571	5678912340
4627530981	3102684597	5610473829	7192034658	6789523401
0576148329	9871546302	9148625307	6219405783	0195634782
1730689452	8956307214	2795380164	3471258906	1956740823
7093812645	5643820971	3986512740	5027143869	9567801234

15. [*50*] Find three 10×10 latin squares that are mutually orthogonal to each other.

16. [*48*] (H. J. Ryser, 1967.) A latin square is said to be of "order n" if it has n rows, n columns, and n symbols. Does every latin square of odd order have a transversal?

17. [*25*] Let L be a latin square with elements L_{ij} for $0 \le i, j < n$. Show that the problems of (a) finding all the transversals of L, and (b) finding all the orthogonal mates of L, are special cases of the general exact cover problem.

18. [*M23*] The string $x_1x_2 \ldots x_N$ is called "n-ary" if each element x_j belongs to the set $\{0, 1, \ldots, n-1\}$ of n-ary digits. Two strings $x_1x_2 \ldots x_N$ and $y_1y_2 \ldots y_N$ are said to be *orthogonal* if the N pairs (x_j, y_j) are distinct for $1 \le j \le N$. (Consequently, two n-ary strings cannot be orthogonal if their length N exceeds n^2.) An n-ary matrix with m rows and n^2 columns whose rows are orthogonal to each other is called an *orthogonal array* of order n and depth m.

Find a correspondence between orthogonal arrays of depth m and lists of $m - 2$ mutually orthogonal latin squares. What orthogonal array corresponds to exercise 11?

▶ **19.** [*M25*] Continuing exercise 18, prove that an orthogonal array of order $n > 1$ and depth m is possible only if $m \le n + 1$. Show that this upper limit is achievable when n is a prime number p. Write out an example when $p = 5$.

20. [*HM20*] Show that if each element k in an orthogonal array is replaced by $e^{2\pi ki/n}$, the rows become orthogonal vectors in the usual sense (their dot product is zero).

▶ **21.** [*M21*] A *geometric net* is a system of points and lines that obeys three axioms:

i) Each line is a set of points.
ii) Distinct lines have at most one point in common.
iii) If p is a point and L is a line with $p \notin L$, then there is exactly one line M such that $p \in M$ and $L \cap M = \emptyset$.

If $L \cap M = \emptyset$ we say that L is *parallel* to M, and write $L \parallel M$.

a) Prove that the lines of a geometric net can be partitioned into equivalence classes, with two lines in the same class if and only if they are equal or parallel.
b) Show that if there are at least two classes of parallel lines, every line contains the same number of points as the other lines in its class.
c) Furthermore, if there are at least three classes, there are numbers m and n such that all points belong to exactly m lines and all lines contain exactly n points.

▶ **22.** [*M22*] Show that every orthogonal array can be regarded as a geometric net. Is the converse also true?

23. [*M21*] (*Error-correcting codes.*) The "Hamming distance" $d(x, y)$ between two strings $x = x_1 \ldots x_N$ and $y = y_1 \ldots y_N$ is the number of positions j where $x_j \ne y_j$. A

b-ary code with n information digits and r check digits is a set $C(b,n,r)$ of b^n strings $x = x_1 \ldots x_{n+r}$, where $0 \le x_j < b$ for $1 \le j \le n+r$. When a codeword x is transmitted and the message y is received, $d(x,y)$ is the number of transmission errors. The code is called *t-error correcting* if we can reconstruct the value of x whenever a message y is received with $d(x,y) \le t$. The *distance* of the code is the minimum value of $d(x,x')$, taken over all pairs of codewords $x \ne x'$.

a) Prove that a code is t-error correcting if and only if its distance exceeds $2t$.
b) Prove that a single-error correcting b-ary code with 2 information digits and 2 check digits is equivalent to a pair of orthogonal latin squares of order b.
c) Furthermore, a code $C(b,2,r)$ with distance $r+1$ is equivalent to a set of r mutually orthogonal latin squares of order b.

▶ **24.** [*M30*] A geometric net with N points and R lines leads naturally to the binary code $C(2,N,R)$ with codewords $x_1 \ldots x_N x_{N+1} \ldots x_{N+R}$ defined by the parity bits

$$x_{N+k} = f_k(x_1,\ldots,x_N) = \bigl(\textstyle\sum\{x_j \mid \text{point } j \text{ lies on line } k\}\bigr) \bmod 2.$$

a) If the net has m classes of parallel lines, prove that this code has distance $m+1$.
b) Find an efficient way to correct up to t errors with this code, assuming that $m = 2t$. Illustrate the decoding process in the case $N = 25$, $R = 30$, $t = 3$.

25. [*27*] Find a latin square whose rows and columns are five-letter words. (For this exercise you'll need to dig out the big dictionaries.)

▶ **26.** [*25*] Compose a meaningful English sentence that contains only five-letter words.

27. [*20*] How many SGB words contain exactly k distinct letters, for $1 \le k \le 5$?

28. [*20*] Are there any pairs of SGB word vectors that differ by ± 1 in each component?

29. [*20*] Find all SGB words that are *palindromes* (equal to their reflection), or mirror pairs (like `regal lager`).

▶ **30.** [*20*] The letters of `first` are in alphabetic order from left to right. What is the lexicographically *first* such five-letter word? What is the last?

31. [*21*] (C. McManus.) Find all sets of three SGB words that are in arithmetic progression but have no common letters in any fixed position. (One such example is {`power`, `slugs`, `visit`}.)

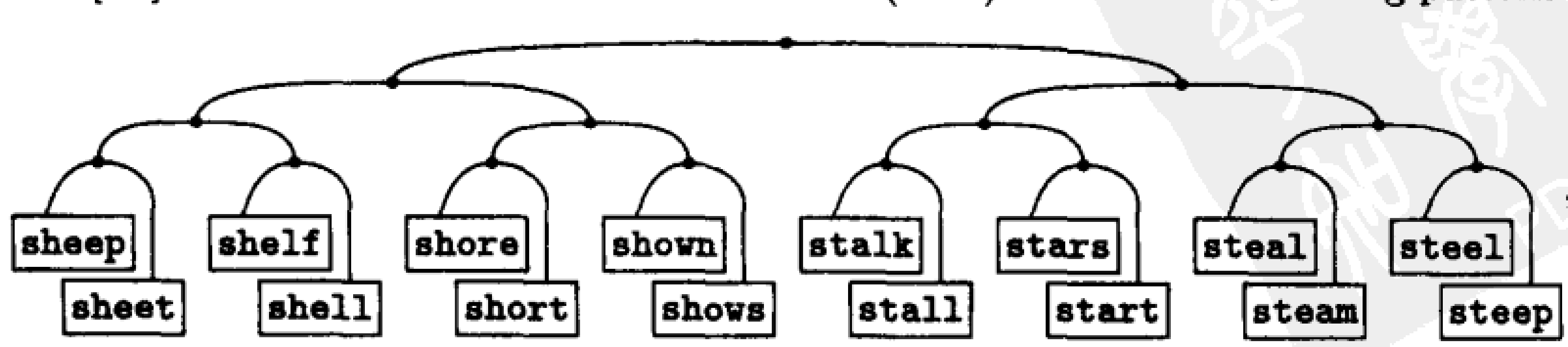
32. [*23*] Does the English language contain any 10-letter words $a_0 a_1 \ldots a_9$ for which both $a_0 a_2 a_4 a_6 a_8$ and $a_1 a_3 a_5 a_7 a_9$ are SGB words?

33. [*20*] (Scot Morris.) Complete the following list of 26 interesting SGB words:

`about`, `bacon`, `faced`, `under`, `chief`, ..., `pizza`.

▶ **34.** [*21*] For each SGB word that doesn't include the letter `y`, obtain a 5-bit binary number by changing the vowels {`a`, `e`, `i`, `o`, `u`} to 1 and the other letters to 0. What are the most common words for each of the 32 binary outcomes?

▶ **35.** [*26*] Sixteen well-chosen elements of `WORDS`(1000) lead to the branching pattern



,

which is a complete binary trie of words that begin with the letter `s`. But there's no such pattern of words beginning with `a`, even if we consider the full collection `WORDS`(5757).

What letters of the alphabet can be used as the starting letter of sixteen words that form a complete binary trie within `WORDS`(n), given n?

36. [*M17*] Explain the symmetries that appear in the word cube (10). Also show that two more such cubes can be obtained by changing only the two words {`stove`, `event`}.

37. [*20*] Which vertices of the graph *words*(5757, 0, 0, 0) have maximum degree?

38. [*22*] Using the digraph rule in (14), change `tears` to `smile` in just three steps, *without computer assistance.*

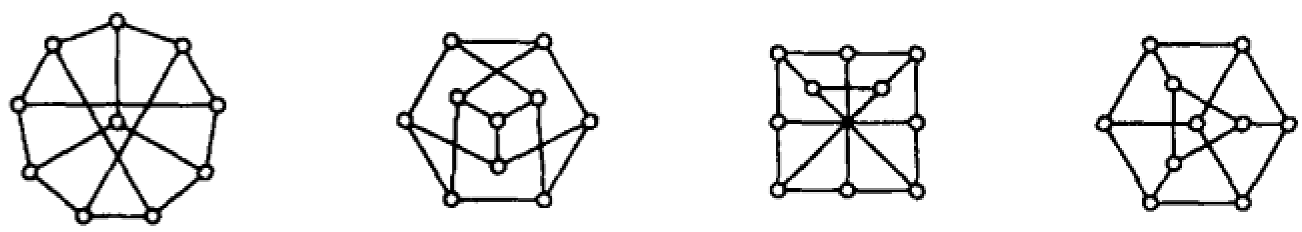
39. [*M00*] Is $G \setminus e$ an induced subgraph of G? Is it a spanning subgraph?

40. [*M15*] How many (a) spanning (b) induced subgraphs does a graph $G = (V, E)$ have, when $|V| = n$ and $|E| = e$?

41. [*M10*] For which integers n do we have (a) $K_n = P_n$? (b) $K_n = C_n$?

42. [*15*] (D. H. Lehmer.) Let G be a graph with 13 vertices, in which every vertex has degree 5. Make a nontrivial statement about G.

43. [*23*] Are any of the following graphs the same as the Petersen graph?



44. [*M23*] How many symmetries does Chvátal's graph have? (See Fig. 2(f).)

45. [*20*] Find an easy way to 4-color the planar graph (17). Would 3 colors suffice?

46. [*M25*] Let G be a graph with $n \geq 3$ vertices, defined by a planar diagram that is "maximal," in the sense that no additional lines can be drawn between nonadjacent vertices without crossing an existing edge.

a) Prove that the diagram partitions the plane into regions that each have exactly three vertices on their boundary. (One of these regions is the set of all points that lie outside the diagram.)

b) Therefore G has exactly $3n - 6$ edges.

47. [*M22*] Prove that the complete bigraph $K_{3,3}$ isn't planar.

48. [*M25*] Complete the proof of Theorem B by showing that the stated procedure never gives the same color to two adjacent vertices.

49. [*18*] Draw diagrams of all the cubic graphs with at most 6 vertices.

50. [*M24*] Find all bipartite graphs that can be 3-colored in exactly 24 ways.

▸ **51.** [*M22*] Given a geometric net as described in exercise 21, construct the bipartite graph whose vertices are the points p and the lines L of the net, with p — L if and only if $p \in L$. What is the *girth* of this graph?

52. [*M16*] Find a simple inequality that relates the diameter of a graph to its girth. (How small can the diameter be, if the girth is large?)

53. [*15*] Which of the words `world` and `happy` belongs to the giant component of the graph *words*(5757, 0, 0, 0)?

▶ **54.** [*21*] The 49 postal codes in graph (17) are AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, IA, ID, IL, IN, KS, KY, LA, MA, MD, ME, MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ, NM, NV, NY, OH, OK, OR, PA, RI, SC, SD, TN, TX, UT, VA, VT, WA, WI, WV, WY, in alphabetical order.

a) Suppose we consider two states to be adjacent if their postal codes agree in one place (namely AL — AR — OR — OH, etc.). What are the components of this graph?

b) Now form a directed graph with XY $\longrightarrow$ YZ (for example, AL $\longrightarrow$ LA $\longrightarrow$ AR, etc.). What are the *strongly connected components* of this digraph? (See Section 2.3.4.2.)

c) The United States has additional postal codes AA, AE, AK, AP, AS, FM, GU, HI, MH, MP, PW, PR, VI, besides those in (17). Reconsider question (b), using all 62 codes.

55. [*M20*] How many edges are in the complete k-partite graph $K_{n_1,\ldots,n_k}$?

▶ **56.** [*M10*] True or false: A multigraph is a graph if and only if the corresponding digraph is simple.

57. [*M10*] True or false: Vertices u and v are in the same connected component of a directed graph if and only if either $d(u,v) < \infty$ or $d(v,u) < \infty$.

58. [*M17*] Describe all (a) graphs (b) multigraphs that are regular of degree 2.

▶ **59.** [*M23*] A *tournament* of order n is a digraph on n vertices that has exactly $\binom{n}{2}$ arcs, either $u \longrightarrow v$ or $v \longrightarrow u$ for every pair of distinct vertices $\{u, v\}$.

a) Prove that every tournament contains an oriented spanning path $v_1 \longrightarrow \cdots \longrightarrow v_n$.

b) Consider the tournament on vertices $\{0,1,2,3,4\}$ for which $u \longrightarrow v$ if and only if $(u-v) \bmod 5 \geq 3$. How many oriented spanning paths does it have?

c) Is $K_n^{\rightarrow}$ the only tournament of order n that has a unique oriented spanning path?

▶ **60.** [*M22*] Let u be a vertex of greatest out-degree in a tournament, and let v be any other vertex. Prove that $d(u,v) \leq 2$.

61. [*M16*] Construct a digraph that has k walks of length k from vertex 1 to vertex 2.

62. [*M21*] A *permutation digraph* is a directed graph in which every vertex has out-degree 1 and in-degree 1; therefore its components are oriented cycles. If it has n vertices and k components, we call it *even* if $n-k$ is even, *odd* if $n-k$ is odd.

a) Let G be a directed graph with adjacency matrix A. Prove that the number of spanning permutation digraphs of G is per A, the permanent of A.

b) Interpret the determinant, det A, in terms of spanning permutation digraphs.

63. [*M23*] Let G be a graph of girth g in which every vertex has at least d neighbors. Prove that G has at least N vertices, where

$$N = \begin{cases} 1 + \sum_{0 \le k < t} d(d-1)^k, & \text{if } g = 2t+1; \\ 1 + (d-1)^t + \sum_{0 \le k < t} d(d-1)^k, & \text{if } g = 2t+2. \end{cases}$$

▶ **64.** [*M21*] Continuing exercise 63, show that there's a *unique* graph of girth 4, minimum degree d, and order $2d$, for each $d \geq 2$.

▶ **65.** [*HM31*] Suppose graph G has girth 5, minimum degree d, and $N = d^2+1$ vertices.

a) Prove that the adjacency matrix A of G satisfies the equation $A^2 + A = (d-1)I + J$.

b) Since A is a symmetric matrix, it has N orthogonal eigenvectors x_j, with corresponding eigenvalues λ_j, such that $Ax_j = \lambda_j x_j$ for $1 \leq j \leq N$. Prove that each λ_j is either d or $(-1 \pm \sqrt{4d-3})/2$.

c) Show that if $\sqrt{4d-3}$ is irrational, then $d = 2$. *Hint:* $\lambda_1 + \cdots + \lambda_N = \text{trace}(A) = 0$.

d) And if $\sqrt{4d-3}$ is rational, $d \in \{3, 7, 57\}$.

66. [*M30*] Continuing exercise 65, construct such a graph when $d = 7$.

67. [*M48*] Is there a regular graph of degree 57, order 3250, and girth 5?

68. [*M20*] How many different adjacency matrices does a graph G on n vertices have?

▸ **69.** [*20*] Extending (31), explain how to calculate both out-degree ODEG(v) and in-degree IDEG(v) for *all* vertices v in a graph that has been represented in SGB format.

▸ **70.** [*M20*] How often is each step of Algorithm B performed, when that algorithm successfully 2-colors a graph with m arcs and n vertices?

71. [*26*] Implement Algorithm B for the MMIX computer, using the MMIXAL assembly language. Assume that, when your program begins, register v0 points to the first vertex node and register n contains the number of vertices.

▸ **72.** [*M22*] When COLOR(v) is set in step B6, call u the *parent* of v; but when COLOR(w) is set in step B3, say that w has no parent. Define the *ancestors* of vertex v, recursively, to be v together with the ancestors of v's parent (if any).

a) Prove that if v is below u in the stack during Algorithm B, the parent of v is an ancestor of u.
b) Furthermore, if COLOR(v) = COLOR(u) in step B6, v is currently in the stack.
c) Use these facts to extend Algorithm B so that, if the given graph is not bipartite, the names of vertices in a cycle of odd length are output.

73. [*15*] What's another name for *random_graph*(10, 45, 0, 0, 0, 0, 0, 0, 0, 0)?

74. [*21*] What vertex of *roget*(1022, 0, 0, 0) has the largest out-degree?

75. [*22*] The SGB graph generator $board(n_1, n_2, n_3, n_4, p, w, o)$ creates a graph whose vertices are the t-dimensional integer vectors $(x_1, \ldots, x_t)$ for $0 \le x_i < b_i$, determined by the first four parameters (n_1, n_2, n_3, n_4) as follows: Set $n_5 \leftarrow 0$ and let $j \ge 0$ be minimum such that $n_{j+1} \le 0$. If $j = 0$, set $b_1 \leftarrow b_2 \leftarrow 8$ and $t \leftarrow 2$; this is the default 8×8 board. Otherwise if $n_{j+1} = 0$, set $b_i \leftarrow n_i$ for $1 \le i \le j$ and $t = j$. Finally, if $n_{j+1} < 0$, set $t \leftarrow |n_{j+1}|$, and set b_i to the ith element of the periodic sequence $(n_1, \ldots, n_j, n_1, \ldots, n_j, n_1, \ldots)$. (For example, the specification $(n_1, n_2, n_3, n_4) = (2, 3, 5, -7)$ is about as tricky as you can get; it produces a 7-dimensional board with $(b_1, \ldots, b_7) = (2, 3, 5, 2, 3, 5, 2)$, hence a graph with $2 \cdot 3 \cdot 5 \cdot 2 \cdot 3 \cdot 5 \cdot 2 = 1800$ vertices.)

The remaining parameters (p, w, o), for "piece, wrap, and orientation," determine the arcs of the graph. Suppose first that $w = o = 0$. If $p > 0$, we have $(x_1, \ldots, x_t) \longrightarrow (y_1, \ldots, y_t)$ if and only if $y_i = x_i + \delta_i$ for $1 \le i \le t$, where $(\delta_1, \ldots, \delta_t)$ is an integer solution to the equation $\delta_1^2 + \cdots + \delta_t^2 = |p|$. And if $p < 0$, we allow also $y_i = x_i + k\delta_i$ for $k \ge 1$, corresponding to k moves in the same direction.

If $w \ne 0$, let $w = (w_t \ldots w_1)_2$ in binary notation. Then we allow "wraparound," $y_i = (x_i + \delta_i) \bmod b_i$ or $y_i = (x_i + k\delta_i) \bmod b_i$, in each coordinate i for which $w_i = 1$.

If $o \ne 0$, the graph is directed; offsets $(\delta_1, \ldots, \delta_t)$ produce arcs only when they are lexicographically greater than $(0, \ldots, 0)$. But if $o = 0$, the graph is undirected.

Find settings of $(n_1, n_2, n_3, n_4, p, w, o)$ for which *board* will produce the following fundamental graphs: (a) the complete graph K_n; (b) the path P_n; (c) the cycle C_n; (d) the transitive tournament $K_n^{\rightarrow}$; (e) the oriented path $P_n^{\rightarrow}$; (f) the oriented cycle $C_n^{\rightarrow}$; (g) the $m \times n$ grid $P_m \,\square\, P_n$; (h) the $m \times n$ cylinder $P_m \,\square\, C_n$; (i) the $m \times n$ torus $C_m \,\square\, C_n$; (j) the $m \times n$ rook graph $K_m \,\square\, K_n$; (k) the $m \times n$ directed torus $C_m^{\rightarrow} \,\square\, C_n^{\rightarrow}$; (l) the null graph $\overline{K_n}$; (m) the n-cube $P_2 \,\square \cdots \square\, P_2$ with 2^n vertices.

76. [*20*] Can $board(n_1, n_2, n_3, n_4, p, w, o)$ produce loops, or parallel (repeated) edges?

77. [*M20*] If graph G has diameter ≥ 3, prove that $\overline{G}$ has diameter ≤ 3.

78. [*M26*] Let $G = (V, E)$ be a graph with $|V| = n$ and $G \cong \overline{G}$. (In other words, G is *self-complementary*: There's a permutation φ of V such that $u \mathbin{—} v$ if and only if $\varphi(u) \not\mathbin{—} \varphi(v)$ and $u \neq v$. We can imagine that the edges of K_n have been painted black or white; the white edges define a graph that's isomorphic to the graph of black edges.)

a) Prove that $n \bmod 4 = 0$ or 1. Draw diagrams for all such graphs with $n < 8$.
b) Prove that if $n \bmod 4 = 0$, every cycle of the permutation φ has a length that is a multiple of 4.
c) Conversely, every permutation φ with such cycles arises in some such graph G.
d) Extend these results to the case $n \bmod 4 = 1$.

▸ **79.** [*M22*] Given $k \geq 0$, construct a graph on the vertices $\{0, 1, \ldots, 4k\}$ that is both regular and self-complementary.

▸ **80.** [*M22*] A self-complementary graph must have diameter 2 or 3, by exercise 77. Given $k \geq 2$, construct self-complementary graphs of both possible diameters, when (a) $V = \{1, 2, \ldots, 4k\}$; (b) $V = \{0, 1, 2, \ldots, 4k\}$.

81. [*20*] The complement of a simple digraph without loops is defined by extending (35) and (36), so that we have $u \to v$ in $\overline{D}$ if and only if $u \neq v$ and $u \not\to v$ in D. What are the self-complementary digraphs of order 3?

82. [*M21*] Are the following statements about line graphs true or false?

a) If G is contained in G', then $L(G)$ is an induced subgraph of $L(G')$.
b) If G is a regular graph, so is $L(G)$.
c) $L(K_{m,n})$ is regular, for all $m, n > 0$.
d) $L(K_{m,n,r})$ is regular, for all $m, n, r > 0$.
e) $L(K_{m,n}) \cong K_m \mathbin{\square} K_n$.
f) $L(K_4) \cong K_{2,2,2}$.
g) $L(P_{n+1}) \cong P_n$.
h) The graphs G and $L(G)$ both have the same number of components.

83. [*16*] Draw the graph $\overline{L(K_5)}$.

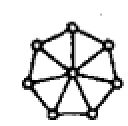
▸ **84.** [*M21*] Is $L(K_{3,3})$ self-complementary?

85. [*M22*] (O. Ore, 1962.) For which graphs G do we have $G \cong L(G)$?

86. [*M20*] (R. J. Wilson.) Find a graph G of order 6 for which $\overline{G} \cong L(G)$.

87. [*20*] Is the Petersen graph (a) 3-colorable? (b) 3-edge-colorable?

88. [*M20*] The graph $W_n = K_1 \mathbin{—} C_{n-1}$ is called the *wheel* of order n, when $n \geq 4$. How many cycles does it contain as subgraphs?



W_8

89. [*M20*] Prove the associative laws, (42) and (43).

▸ **90.** [*M24*] A graph is called a *cograph* if it can be constructed algebraically from 1-element graphs by means of complementation and/or direct sum operations. For example, there are four nonisomorphic graphs of order 3, and they all are cographs: $\overline{K_3} = K_1 \oplus K_1 \oplus K_1$ and its complement, K_3; $\overline{K_{1,2}} = K_1 \oplus K_2$ and its complement, $K_{1,2}$, where $K_2 = \overline{K_1 \oplus K_1}$.

Exhaustive enumeration shows that there are 11 nonisomorphic graphs of order 4. Give algebraic formulas to prove that 10 of them are cographs. Which one isn't?

▸ **91.** [*20*] Draw diagrams for the 4-vertex graphs (a) $K_2 \mathbin{\square} K_2$; (b) $K_2 \otimes K_2$; (c) $K_2 \boxtimes K_2$; (d) $K_2 \mathbin{\triangle} K_2$; (e) $K_2 \circ K_2$; (f) $\overline{K_2} \circ K_2$; (g) $K_2 \circ \overline{K_2}$.

92. [*21*] The five types of graph products defined in the text work fine for simple digraphs as well as for ordinary graphs. Draw diagrams for the 4-vertex digraphs (a) $\vec{K_2} \mathbin{\square} \vec{K_2}$; (b) $\vec{K_2} \otimes \vec{K_2}$; (c) $\vec{K_2} \boxtimes \vec{K_2}$; (d) $\vec{K_2} \mathbin{\triangle} \vec{K_2}$; (e) $\vec{K_2} \circ \vec{K_2}$.

93. [*15*] Which of the five graph products takes K_m and K_n into K_{mn}?

94. [*10*] Are the SGB *words* graphs induced subgraphs of $P_{26} \square P_{26} \square P_{26} \square P_{26} \square P_{26}$?

95. [*M20*] If vertex u of G has degree d_u and vertex v of H has degree d_v, what is the degree of vertex (u, v) in (a) $G \square H$? (b) $G \otimes H$? (c) $G \boxtimes H$? (d) $G \triangle H$? (e) $G \circ H$?

▸ **96.** [*M22*] Let A be an $m \times m'$ matrix with $a_{uu'}$ in row u and column u'; let B be an $n \times n'$ matrix with $b_{vv'}$ in row v and column v'. The *direct product* $A \otimes B$ is an $mn \times m'n'$ matrix with $a_{uu'}b_{vv'}$ in row (u, v) and column (u', v'). Thus $A \otimes B$ is the adjacency matrix of $G \otimes H$, if A and B are the adjacency matrices of G and H.

Find analogous formulas for the adjacency matrices of (a) $G \square H$; (b) $G \boxtimes H$; (c) $G \triangle H$; (d) $G \circ H$.

97. [*M25*] Find as many interesting algebraic relations between graph sums and products as you can. (For example, the distributive law $(A \oplus B) \otimes C = (A \otimes C) \oplus (B \otimes C)$ for direct sums and products of matrices implies that $(G \oplus G') \otimes H = (G \otimes H) \oplus (G' \otimes H)$. We also have $\overline{K_m} \square H = H \oplus \cdots \oplus H$, with m copies of H, etc.)

98. [*M20*] If the graph G has k components and the graph H has l components, how many components are in the graphs $G \square H$ and $G \boxtimes H$?

99. [*M20*] Let $d_G(u, u')$ be the distance from vertex u to vertex u' in graph G. Prove that $d_{G \square H}((u, v), (u', v')) = d_G(u, u') + d_H(v, v')$, and find a similar formula for $d_{G \boxtimes H}((u, v), (u', v'))$.

100. [*M21*] For which connected graphs is $G \otimes H$ connected?

▸ **101.** [*M25*] Find all connected graphs G and H such that $G \square H \cong G \otimes H$.

102. [*M20*] What's a simple algebraic formula for the graph of *king moves* (which take one step horizontally, vertically, or diagonally) on an $m \times n$ board?

103. [*20*] Complete tableau (54). Also apply Algorithm H to the sequence 866444444.

104. [*18*] Explain the manipulation of variables i, t, and r in steps H3 and H4.

105. [*M34*] Suppose $d_1 \geq \cdots \geq d_n \geq 0$, and let $c_1 \geq \cdots \geq c_{d_1}$ be its conjugate as in Algorithm H. Prove that $d_1 \ldots d_n$ is graphical if and only if $d_1 + \cdots + d_n$ is even and $d_1 + \cdots + d_k \leq c_1 + \cdots + c_k - k$ for $1 \leq k \leq s$, where s is maximal such that $d_s \geq s$.

106. [*20*] True or false: If $d_1 = \cdots = d_n = d < n$ and nd is even, Algorithm H constructs a *connected* graph.

107. [*M21*] Prove that the degree sequence $d_1 \ldots d_n$ of a self-complementary graph satisfies $d_j + d_{n+1-j} = n - 1$ and $d_{2j-1} = d_{2j}$ for $1 \leq j \leq n/2$.

▸ **108.** [*M23*] Design an algorithm analogous to Algorithm H that constructs a *simple directed graph* on vertices $\{1, \ldots, n\}$, having specified values d_k^- and d_k^+ for the in-degree and out-degree of each vertex k, whenever at least one such graph exists.

109. [*M20*] Design an algorithm analogous to Algorithm H that constructs a *bipartite graph* on vertices $\{1, \ldots, m + n\}$, having specified degrees d_k for each vertex k when possible; all edges $j — k$ should have $j \leq m$ and $k > m$.

110. [*M22*] Without using Algorithm H, show by a direct construction that the sequence $d_1 \ldots d_n$ is graphical when $n > d_1 \geq \cdots \geq d_n \geq d_1 - 1$ and $d_1 + \cdots + d_n$ is even.

▸ **111.** [*25*] Let G be a graph on vertices $V = \{1, \ldots, n\}$, with d_k the degree of k and $\max(d_1, \ldots, d_n) = d$. Prove that there's an integer N with $n \leq N \leq 2n$ and a graph H on vertices $\{1, \ldots, N\}$, such that H is regular of degree d and $H \mid V = G$. Explain how to construct such a regular graph with N as small as possible.

▶ **112.** [*20*] Does the network *miles*(128, 0, 0, 0, 0, 127, 0) have three equidistant cities? If not, what three cities come closest to an equilateral triangle?

113. [*05*] When H is a hypergraph with m edges and n vertices, how many rows and columns does its incidence matrix have?

114. [*M20*] Suppose the multigraph (26) is regarded as a hypergraph. What is the corresponding incidence matrix? What is the corresponding bipartite graph?

▶ **115.** [*M20*] When B is the incidence matrix of a graph G, explain the significance of the symmetric matrices B^TB and BB^T.

116. [*M17*] Describe the edges of the complete bipartite r-uniform hypergraph $K^{(r)}_{m,n}$.

117. [*M22*] How many nonisomorphic 1-uniform hypergraphs have m edges and n vertices? (Edges may be repeated.) List them all when $m = 4$ and $n = 3$.

118. [*M20*] A "hyperforest" is a hypergraph that contains no cycles. If a hyperforest has m edges, n vertices, and p components, what's the sum of the degrees of its vertices?

119. [*M18*] What hypergraph corresponds to (60) without the final term $(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$?

120. [*M20*] Define *directed hypergraphs*, by generalizing the concept of directed graphs.

121. [*M19*] Given a hypergraph $H = (V, E)$, let $I(H) = (V, F)$, where F is the family of all maximal independent sets of H. Express $\chi(H)$ in terms of $|V|$, $|F|$, and $\alpha(I(H)^T)$.

▶ **122.** [*M24*] Find a maximum independent set and a minimum coloring of the following triple systems: (a) the hypergraph (56); (b) the dual of the Petersen graph.

123. [*17*] Show that the optimum colorings of $K_n \square K_n$ are equivalent to the solutions of a famous combinatorial problem.

124. [*M22*] What is the chromatic number of the Chvátal graph, Fig. 2(f)?

125. [*M48*] For what values of g is there a 4-regular, 4-chromatic graph of girth g?

▶ **126.** [*M22*] Find optimum colorings of the "kingwise torus," $C_m \boxtimes C_n$, when $m, n \geq 3$.

127. [*M22*] Prove that (a) $\chi(G) + \chi(\overline{G}) \leq n + 1$ and (b) $\chi(G)\chi(\overline{G}) \geq n$ when G is a graph of order n, and find graphs for which equality holds.

128. [*M18*] Express $\chi(G \square H)$ in terms of $\chi(G)$ and $\chi(H)$, when G and H are graphs.

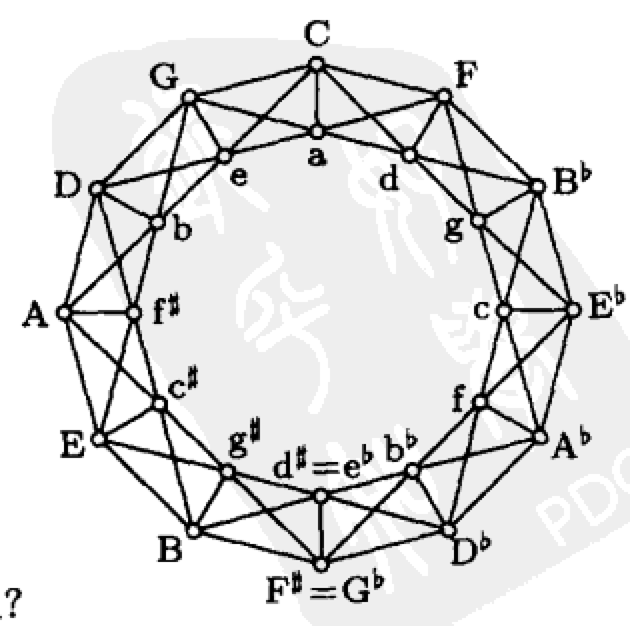
129. [*23*] Describe the maximal cliques of the 8×8 queen graph (37).

130. [*M20*] How many maximal cliques are in a complete k-partite graph?

131. [*M30*] Let $N(n)$ be the largest number of maximal cliques that an n-vertex graph can have. Prove that $3^{\lfloor n/3 \rfloor} \leq N(n) \leq 3^{\lceil n/3 \rceil}$.

▶ **132.** [*M20*] We call G *tightly colorable* if $\chi(G) = \omega(G)$. Prove that $\chi(G \boxtimes H) = \chi(G)\chi(H)$ whenever G and H are tightly colorable.

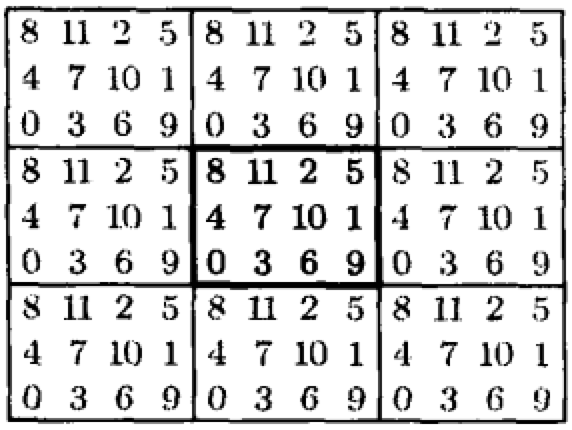
133. [*21*] The "musical graph" illustrated here provides a nice way to review numerous definitions that were given in this section, because its properties are easily analyzed. Determine its (a) order; (b) size; (c) girth; (d) diameter; (e) independence number, $\alpha(G)$; (f) chromatic number, $\chi(G)$; (g) edge-chromatic number, $\chi(L(G))$; (h) clique number, $\omega(G)$; (i) algebraic formula as a product of well-known smaller graphs. What is the size of (j) a minimum vertex cover? (k) a maximum matching? Is G (l) regular? (m) planar? (n) connected? (o) directed? (p) a free tree? (q) Hamiltonian?

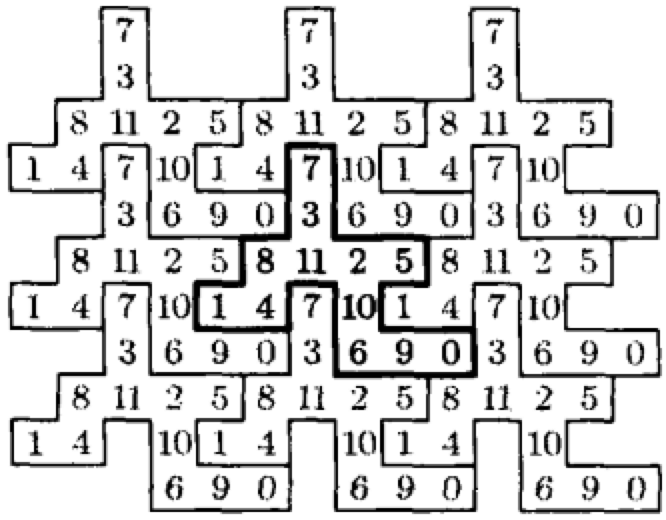


134. [*M22*] How many automorphisms does the musical graph have?

▶ **135.** [*HM26*] Suppose a composer takes a random walk in the musical graph, starting at vertex C and then making five equally likely choices at each step. Show that after an even number of steps, the walk is more likely to end at vertex C than at any other vertex. What is the exact probability of going from C to C in a 12-step walk?

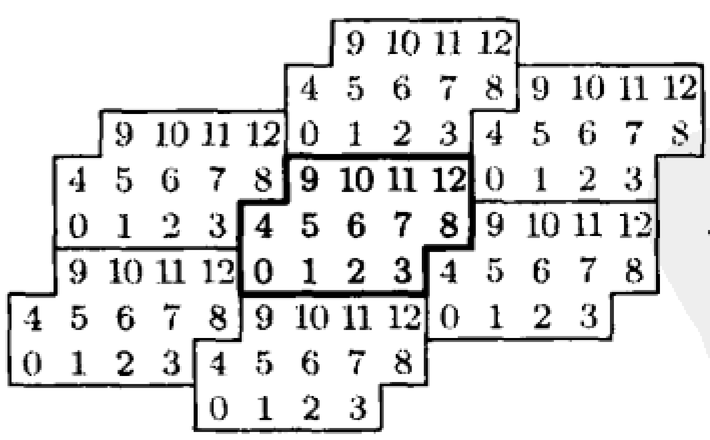
136. [*HM23*] A *Cayley digraph* is a directed graph whose vertices V are the elements of a group and whose arcs are $v \longrightarrow v\alpha_j$ for $1 \le j \le d$ and all vertices v, where $(\alpha_1, \ldots, \alpha_d)$ are fixed elements of the group. A *Cayley graph* is a Cayley digraph that is also a graph. Is the Petersen graph a Cayley graph?





▶ **137.** [*M25*] (*Generalized toruses.*) An $m \times n$ torus can be regarded as a tiling of the plane. For example, we can imagine that infinitely many copies of the 3×4 torus in (50) have been placed together gridwise, as indicated in the left-hand illustration above; from each vertex we can move north, south, east, or west to another vertex of the torus. The vertices have been numbered here so that a northward move from v goes to $(v+4) \bmod 12$, and an eastward move to $(v+3) \bmod 12$, etc. The right-hand illustration shows the same torus, but with a differently shaped tile; *any* way to choose twelve cells numbered $\{0, 1, \ldots, 11\}$ will tile the plane, with exactly the same underlying graph.

Shifted copies of a single shape will also tile the plane if they form a *generalized* torus, in which cell (x, y) corresponds to the same vertex as cells $(x + a, y + b)$ and $(x + c, y + d)$, where (a, b) and (c, d) are integer vectors and $n = ad - bc > 0$. The generalized torus will then have n points. These vectors (a, b) and (c, d) are $(4, 0)$ and $(0, 3)$ in the 3×4 example above; and when they are respectively $(5, 2)$ and $(1, 3)$ we get



Here $n = 13$, and a northward move from v goes to $(v + 4) \bmod 13$; an eastward move goes to $(v + 1) \bmod 13$.

Prove that if $\gcd(a, b, c, d) = 1$, the vertices of such a generalized torus can always be assigned integer labels $\{0, 1, \ldots, n-1\}$ in such a way that the neighbors of v are $(v \pm p) \bmod n$ and $(v \pm q) \bmod n$, for some integers p and q.

138. [*HM27*] Continuing exercise 137, what is a good way to label k-dimensional vertices $x = (x_1, \ldots, x_k)$, when integer vectors α_j are given such that each vector x is equivalent to $x + \alpha_j$ for $1 \le j \le k$? Illustrate your method in the case $k = 3$, $\alpha_1 = (3,1,1)$, $\alpha_2 = (1,3,1)$, $\alpha_3 = (1,1,3)$.

▸ **139.** [*M22*] Let H be a fixed graph of order h, and let $\#(H{:}G)$ be the number of times that H occurs as an induced subgraph of a given graph G. If G is chosen at random from the set of all $2^{n(n-1)/2}$ graphs on the vertices $V = \{1, 2, \ldots, n\}$, what is the average value of $\#(H{:}G)$ when H is (a) K_h; (b) P_h, for $h > 1$; (c) C_h, for $h > 2$; (d) arbitrary?

140. [*M30*] A graph G is called *proportional* if its induced subgraph counts $\#(K_3{:}G)$, $\#(\overline{K_3}{:}G)$, and $\#(P_3{:}G)$ each agree with the expected values derived in exercise 139.

a) Show that the wheel graph W_8 of exercise 88 is proportional in this sense.
b) Prove that G is proportional if and only if $\#(K_3{:}G) = \frac{1}{8}\binom{n}{3}$ and the degree sequence $d_1 \ldots d_n$ of its vertices satisfies the identities

$$d_1 + \cdots + d_n = \binom{n}{2}, \qquad d_1^2 + \cdots + d_n^2 = \frac{n}{2}\binom{n}{2}. \qquad (*)$$

141. [*26*] The conditions of exercise 140(b) can hold only if $n \bmod 16 \in \{0, 1, 8\}$. Write a program to find all of the proportional graphs that have $n = 8$ vertices.

142. [*M30*] (S. Janson and J. Kratochvíl, 1991.) Prove that no graph G on 4 or more vertices can be "extraproportional," in the sense that its subgraph counts $\#(H{:}G)$ agree with the expected values in exercise 139 for each of the eleven nonisomorphic graphs H of order 4. *Hint:* $(n-3)\#(K_3{:}G) = 4\#(K_4{:}G) + 2\#(K_{1,1,2}{:}G) + \#(K_1 \oplus K_3{:}G)$.

▸ **143.** [*M25*] Let A be any matrix with $m > 1$ distinct rows, and $n \ge m$ columns. Prove that at least one column of A can be deleted, without making any two rows equal.

▸ **144.** [*21*] Let X be an $m \times n$ matrix whose entries x_{ij} are either 0, 1, or $*$. A "completion" of X is a matrix X^* in which every $*$ has been replaced by either 0 or 1. Show that the problem of finding a completion with fewest distinct rows is equivalent to the problem of finding the chromatic number of a graph.

▸ **145.** [*25*] (R. S. Boyer and J. S. Moore, 1980.) Suppose the array $a_1 \ldots a_n$ contains a *majority element*, namely a value that occurs more than $n/2$ times. Design an algorithm that finds it after making fewer than n comparisons. *Hint:* If $n \ge 3$ and $a_{n-1} \ne a_n$, the majority element of $a_1 \ldots a_n$ is also the majority element of $a_1 \ldots a_{n-2}$.

Yet now and then your men of wit
Will condescend to take a bit.
— JONATHAN SWIFT, *Cadenus and Vanessa* (1713)

If the base 2 is used the resulting units may be called binary digits,
or more briefly bits, *a word suggested by J. W. Tukey.*
— CLAUDE E. SHANNON, in *Bell System Technical Journal* (1948)

bit (bit), n . . . *[A] boring tool . . .*
— *Random House Dictionary of the English Language* (1987)

7.1. ZEROS AND ONES

COMBINATORIAL ALGORITHMS often require special attention to efficiency, and the proper representation of data is an important way to gain the necessary speed. It is therefore wise to beef up our knowledge of elementary representation techniques before we set out to study combinatorial algorithms in detail.

Most of today's computers are based on the binary number system, instead of working directly with the decimal numbers that human beings prefer, because machines are especially good at dealing with the two-state on-off quantities that we usually denote by the digits 0 and 1. But in Chapters 1 to 6 we haven't made much use of the fact that binary computers can do several things quickly that decimal computers cannot. A binary machine can usually perform "logical" or "bitwise" operations just as easily as it can add or subtract; yet we have seldom capitalized on that capability. We've seen that binary and decimal computers are not significantly different, for many purposes, but in a sense we've been asking a binary computer to operate with one hand tied behind its back.

The amazing ability of 0s and 1s to encode information as well as to encode the logical relations between items, and even to encode algorithms for processing information, makes the study of binary digits especially rich. Indeed, we not only use bitwise operations to enhance combinatorial algorithms, we also find that the properties of binary logic lead naturally to new combinatorial problems that are of great interest in their own right.

Computer scientists have gradually become better and better at taming the wild 0s and 1s of the universe and making them do useful tricks. But as bit players on the world's stage, we'd better have a thorough understanding of the low-level properties of binary quantities before we launch into a study of higher-level concepts and techniques. Therefore we shall start by investigating basic ways to combine individual bits and sequences of bits.

7.1.1. Boolean Basics

There are 16 possible functions $f(x,y)$ that transform two given bits x and y into a third bit $z = f(x,y)$, since there are two choices for each of $f(0,0)$, $f(0,1)$, $f(1,0)$, and $f(1,1)$. Table 1 indicates the names and notations that have traditionally been associated with these functions in studies of formal logic, assuming that 1 corresponds to "true" and 0 to "false." The sequence of four values $f(0,0)f(0,1)f(1,0)f(1,1)$ is customarily called the *truth table* of the function f.

Let us conceive, then, of an Algebra in which the symbols x, y, z, &c. admit indifferently of the values 0 and 1, and of these values alone.
— GEORGE BOOLE, *An Investigation of the Laws of Thought* (1854)

'Contrariwise,' continued Tweedledee, 'if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic.'
— LEWIS CARROLL, *Through the Looking Glass* (1871)

Such functions are often called "Boolean operations" in honor of George Boole, who first discovered that algebraic operations on 0s and 1s could be used to construct a calculus for logical reasoning [*The Mathematical Analysis of Logic* (Cambridge: 1847); *An Investigation of the Laws of Thought* (London: 1854)]. But Boole never actually dealt with the "logical or" operation $\vee$; he confined himself strictly to ordinary arithmetic operations on 0s and 1s. Thus he would write $x+y$ to stand for disjunction, but he took pains never to use this notation unless x and y were mutually exclusive (not both 1). If necessary, he wrote $x+(1-x)y$ to ensure that the result of a disjunction would never be equal to 2.

When rendering the + operation in English, Boole sometimes called it "and," sometimes "or." This practice may seem strange to modern mathematicians until we realize that his usage was in fact normal English; we say, for example, that "boys and girls are children," but "children are boys or girls."

Boole's calculus was extended to include the unconventional rule $x+x=x$ by W. Stanley Jevons [*Pure Logic* (London: Edward Stanford, 1864), §69], who pointed out that $(x+y)z$ was equal to $xz+yz$ using his new + operation. But Jevons did not know the other distributive law $xy+z=(x+z)(y+z)$. Presumably he missed this because of the notation he was using, since the second distributive law has no familiar counterpart in arithmetic; the more symmetrical notations $x\wedge y$, $x\vee y$ in Table 1 make it easier for us to remember both distributive laws

$$(x\vee y)\wedge z \;=\; (x\wedge z)\vee(y\wedge z); \tag{1}$$

$$(x\wedge y)\vee z \;=\; (x\vee z)\wedge(y\vee z). \tag{2}$$

The second law (2) was introduced by C. S. Peirce, who had discovered independently how to extend Boole's calculus [*Proc. Amer. Acad. Arts and Sciences* **7** (1867), 250–261]. Incidentally, when Peirce discussed these early developments several years later [*Amer. J. Math.* **3** (1880), 32], he referred to "the Boolian algebra, with Jevons's addition"; his now-unfamiliar spelling of "Boolean" was in use for many years, appearing in the Funk and Wagnalls unabridged dictionary as late as 1963.

The notion of truth-value combination is actually much older than Boolean algebra. Indeed, propositional logic had been developed by Greek philosophers already in the fourth century B.C. There was considerable debate in those days about how to assign an appropriate true-or-false value to the proposition "if x then y" when x and y are propositions; Philo of Megara, about 300 B.C., defined

Table 1

THE SIXTEEN LOGICAL OPERATIONS ON TWO VARIABLES

Truth table	Notation(s)	Operator symbol $\circ$	Name(s)
0000	0	$\perp$	Contradiction; falsehood; antilogy; constant 0
0001	$xy,\ \ x \land y,\ \ x \mathbin{\&} y$	$\land$	Conjunction; and
0010	$x \land \bar{y},\ \ x \not\supset y,\ \ [x > y],\ \ x \mathbin{\dot-} y$	$\bar{\supset}$	Nonimplication; difference; but not
0011	x	L	Left projection
0100	$\bar{x} \land y,\ \ x \not\subset y,\ \ [x < y],\ \ y \mathbin{\dot-} x$	$\bar{\subset}$	Converse nonimplication; not ... but
0101	y	R	Right projection
0110	$x \oplus y,\ \ x \not\equiv y,\ \ x\,\hat{}\,y$	$\oplus$	Exclusive disjunction; nonequivalence; "xor"
0111	$x \lor y,\ \ x \mid y$	$\lor$	(Inclusive) disjunction; or; and/or
1000	$\bar{x} \land \bar{y},\ \ \overline{x \lor y},\ \ x \mathbin{\bar{\lor}} y,\ \ x \downarrow y$	$\bar{\lor}$	Nondisjunction; joint denial; neither ... nor
1001	$x \equiv y,\ \ x \leftrightarrow y,\ \ x \Leftrightarrow y$	$\equiv$	Equivalence; if and only if; "iff"
1010	$\bar{y},\ \ \neg y,\ \ !y,\ \ \sim y$	$\bar{\mathrm{R}}$	Right complementation
1011	$x \lor \bar{y},\ x \subset y,\ x \Leftarrow y,\ [x \geq y],\ x^y$	$\subset$	Converse implication; if
1100	$\bar{x},\ \ \neg x,\ \ !x,\ \ \sim x$	$\bar{\mathrm{L}}$	Left complementation
1101	$\bar{x} \lor y,\ x \supset y,\ x \Rightarrow y,\ [x \leq y],\ y^x$	$\supset$	Implication; only if; if ... then
1110	$\bar{x} \lor \bar{y},\ \ \overline{x \land y},\ \ x \mathbin{\bar{\land}} y,\ \ x \mid y$	$\bar{\land}$	Nonconjunction; not both ... and; "nand"
1111	1	$\top$	Affirmation; validity; tautology; constant 1

it by the truth table shown in Table 1, which states in particular that the implication is true when both x and y are false. Much of this early work has been lost, but there are passages in the works of Galen (2nd century A.D.) that refer to both inclusive and exclusive disjunction of propositions. [See I. M. Bocheński, *Formale Logik* (1956), English translation by Ivo Thomas (1961), for an excellent survey of the development of logic from ancient times up to the 20th century.]

A function of two variables is often written $x \circ y$ instead of $f(x, y)$, using some appropriate operator symbol $\circ$. Table 1 shows the sixteen operator symbols that we shall adopt for Boolean functions of two variables; for example, $\perp$ symbolizes the function whose truth table is 0000, $\land$ is the symbol for 0001, $\bar{\supset}$ is the symbol for 0010, and so on. We have $x \perp y = 0$, $x \land y = xy$, $x \mathbin{\bar{\supset}} y = x \mathbin{\dot-} y$, $x \mathbin{\mathrm{L}} y = x$, $\ldots$, $x \mathbin{\bar{\land}} y = \bar{x} \lor \bar{y}$, $x \top y = 1$.

Of course the operations in Table 1 aren't all of equal importance. For example, the first and last cases are trivial, since they have a constant value independent of x and y. Four of them are functions of x alone or y alone. We write $\bar{x}$ for $1 - x$, the *complement* of x.

The four operations whose truth table contains just a single 1 are easily expressed in terms of the AND operator $\land$, namely $x \land y$, $x \land \bar{y}$, $\bar{x} \land y$, $\bar{x} \land \bar{y}$. Those with three 1s are easily written in terms of the OR operator $\lor$, namely $x \lor y$, $x \lor \bar{y}$, $\bar{x} \lor y$, $\bar{x} \lor \bar{y}$. The basic functions $x \land y$ and $x \lor y$ have proved to be more useful in practice than their complemented or half-complemented cousins, although the NOR and NAND operations $x \mathbin{\bar{\lor}} y = \bar{x} \land \bar{y}$ and $x \mathbin{\bar{\land}} y = \bar{x} \lor \bar{y}$ are also of interest because they are easily implemented in transistor circuits.

In 1913, H. M. Sheffer showed that all 16 of the functions can be expressed in terms of just one, starting with either $\bar{\vee}$ or $\bar{\wedge}$ as the given operation (see exercise 4). Actually C. S. Peirce had made the same discovery about 1880, but his work on the subject remained unpublished until after his death [*Collected Papers of Charles Sanders Peirce* **4** (1933), §§12–20, 264]. Table 1 indicates that NAND and NOR have occasionally been written $x \mid y$ and $x \downarrow y$; sometimes they have been called "Sheffer's stroke" and the "Peirce arrow." Nowadays it is best *not* to use Sheffer's vertical line for NAND, because $x \mid y$ denotes bitwise $x \vee y$ in programming languages like C.

So far we have discussed all but two of the functions in Table 1. The remaining two are $x \equiv y$ and $x \oplus y$, "equivalence" and "exclusive-or," which are related by the identities

$$x \equiv y \;=\; \bar{x} \oplus y \;=\; x \oplus \bar{y} \;=\; 1 \oplus x \oplus y; \tag{3}$$

$$x \oplus y \;=\; \bar{x} \equiv y \;=\; x \equiv \bar{y} \;=\; 0 \equiv x \equiv y. \tag{4}$$

Both operations are associative (see exercise 6). In propositional logic, the notion of equivalence is more important than the notion of exclusive-or, which means inequivalence; but when we consider bitwise operations on full computer words, we shall see in Section 7.1.3 that the situation is reversed: Exclusive-or turns out to be more useful than equivalence, in typical programs. The chief reason why $x \oplus y$ has significant applications, even in the one-bit case, is the fact that

$$x \oplus y \;=\; (x + y) \bmod 2. \tag{5}$$

Therefore $x \oplus y$ and $x \wedge y$ denote addition and multiplication in the field of two elements (see Section 4.6), and $x \oplus y$ naturally inherits many "clean" mathematical properties.

Basic identities. Now let's take a look at interactions between the fundamental operators $\wedge$, $\vee$, $\oplus$, and $^{-}$, since the other operations are easily expressed in terms of these four. Each of $\wedge$, $\vee$, $\oplus$ is associative and commutative. Besides the distributive laws (1) and (2), we also have

$$(x \oplus y) \wedge z \;=\; (x \wedge z) \oplus (y \wedge z), \tag{6}$$

as well as the *absorption laws*

$$(x \wedge y) \vee x \;=\; (x \vee y) \wedge x \;=\; x. \tag{7}$$

One of the simplest, yet most useful, identities is

$$x \oplus x \;=\; 0, \tag{8}$$

since it implies among other things that

$$(x \oplus y) \oplus x \;=\; y, \qquad (x \oplus y) \oplus y \;=\; x, \tag{9}$$

when we use the obvious fact that $x \oplus 0 = x$. In other words, given $x \oplus y$ and either x or y, it is easy to determine the other. And let us not overlook the simple *complementation law*

$$\bar{x} \;=\; x \oplus 1. \tag{10}$$

Another important pair of identities is known as *De Morgan's laws* in honor of Augustus De Morgan, who stated that "The contrary of an aggregate is the compound of the contraries of the aggregants; the contrary of a compound is the aggregate of the contraries of the components. Thus (A, B) and AB have ab and (a, b) for contraries." [*Trans. Cambridge Philos. Soc.* **10** (1858), 208.] In more modern notation, these are the rules we have implicitly derived via truth tables in connection with the operations NAND and NOR in Table 1, namely

$$\overline{x \wedge y} \;=\; \bar{x} \vee \bar{y}; \tag{11}$$
$$\overline{x \vee y} \;=\; \bar{x} \wedge \bar{y}. \tag{12}$$

Incidentally, W. S. Jevons knew (12) but not (11); he consistently wrote $\bar{A}B + \bar{B}A + \bar{A}\bar{B}$ instead of $\bar{A} + \bar{B}$ for the complement of AB. Yet De Morgan was not the first Englishman who enunciated the laws above. Both (11) and (12) can be found in the early 14th century writings of two scholastic philosophers, William of Ockham [*Summa Logicæ* **2** (1323)] and Walter Burley [*De Puritate Artis Logicæ* (c. 1330)].

De Morgan's laws and a few other identities can be used to express $\wedge$, $\vee$, and $\oplus$ in terms of each other:

$$x \wedge y \;=\; \overline{\bar{x} \vee \bar{y}} \;=\; x \oplus y \oplus (x \vee y); \tag{13}$$
$$x \vee y \;=\; \overline{\bar{x} \wedge \bar{y}} \;=\; x \oplus y \oplus (x \wedge y); \tag{14}$$
$$x \oplus y \;=\; (x \vee y) \wedge \overline{x \wedge y} \;=\; (x \wedge \bar{y}) \vee (\bar{x} \wedge y). \tag{15}$$

According to exercise 7.1.2–77, all computations of $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ that use only the operations $\wedge$, $\vee$, and $^-$ must be at least $4(n-1)$ steps long; thus, the other three operations are not an especially good substitute for $\oplus$.

Functions of n variables. A Boolean function $f(x, y, z)$ of three Boolean variables x, y, z can be defined by its 8-bit truth table $f(0,0,0)f(0,0,1)\ldots f(1,1,1)$; and in general, every n-ary Boolean function $f(x_1, \ldots, x_n)$ corresponds to a 2^n-bit truth table that lists the successive values of $f(0, \ldots, 0, 0)$, $f(0, \ldots, 0, 1)$, $f(0, \ldots, 1, 0)$, …, $f(1, \ldots, 1, 1)$.

We needn't devise special names and notations for all these functions, since they can all be expressed in terms of the binary functions that we've already learned. For example, as observed by I. I. Zhegalkin [*Matematicheskiĭ Sbornik* **35** (1928), 311–369], we can always write

$$f(x_1, \ldots, x_n) \;=\; g(x_1, \ldots, x_{n-1}) \;\oplus\; h(x_1, \ldots, x_{n-1}) \wedge x_n \tag{16}$$

when $n > 0$, for appropriate functions g and h, by letting

$$\begin{aligned} g(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 0); \\ h(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 0) \oplus f(x_1, \ldots, x_{n-1}, 1). \end{aligned} \tag{17}$$

(The operation $\wedge$ conventionally takes precedence over $\oplus$, so we need not use parentheses to enclose the subformula '$h(x_1, \ldots, x_{n-1}) \wedge x_n$' on the right-hand side of (16).) Repeating this process recursively on g and h until we're down to

0-ary functions leaves us with an expression that involves only the operators $\oplus$, $\wedge$, and a sequence of 2^n constants. Furthermore, those constants can usually be simplified away, because we have

$$x \wedge 0 = 0 \qquad \text{and} \qquad x \wedge 1 = x \oplus 0 = x. \tag{18}$$

After applying the associative and distributive laws, we end up needing the constant 0 only if $f(x_1, \ldots, x_n)$ is identically zero, and the constant 1 only if $f(0, \ldots, 0) = 1$.

We might have, for instance,

$$\begin{aligned} f(x,y,z) &= \bigl((1 \oplus 0 \wedge x) \oplus (0 \oplus 1 \wedge x) \wedge y\bigr) \oplus \bigl((0 \oplus 1 \wedge x) \oplus (1 \oplus 1 \wedge x) \wedge y\bigr) \wedge z \\ &= (1 \oplus x \wedge y) \;\oplus\; (x \oplus y \oplus x \wedge y) \wedge z \\ &= 1 \;\oplus\; x \wedge y \;\oplus\; x \wedge z \;\oplus\; y \wedge z \;\oplus\; x \wedge y \wedge z. \end{aligned}$$

And by rule (5), we see that we're simply left with the polynomial

$$f(x,y,z) \;=\; (1 + xy + xz + yz + xyz) \bmod 2, \tag{19}$$

because $x \wedge y = xy$. Notice that this polynomial is linear (of degree ≤ 1) in each of its variables. In general, a similar calculation will show that *any* Boolean function $f(x_1, \ldots, x_n)$ has a unique representation such as this, called its *multilinear representation*, which is a sum (modulo 2) of zero or more of the 2^n possible terms 1, x_1, x_2, x_1x_2, x_3, x_1x_3, x_2x_3, $x_1x_2x_3$, $\ldots$, $x_1x_2\ldots x_n$.

George Boole decomposed Boolean functions in a different way, which is often simpler for the kinds of functions that arise in practice. Instead of (16), he essentially wrote

$$f(x_1, \ldots, x_n) \;=\; \bigl(g(x_1, \ldots, x_{n-1}) \wedge \bar{x}_n\bigr) \;\vee\; \bigl(h(x_1, \ldots, x_{n-1}) \wedge x_n\bigr) \tag{20}$$

and called it the "law of development," where we now have simply

$$\begin{aligned} g(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 0), \\ h(x_1, \ldots, x_{n-1}) &= f(x_1, \ldots, x_{n-1}, 1), \end{aligned} \tag{21}$$

instead of (17). Repeatedly iterating Boole's procedure, using the distributive law (1), and eliminating constants, leaves us with a formula that is a disjunction of zero or more *minterms*, where each minterm is a conjunction such as $x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge x_5$ in which every variable or its complement is present. Notice that a minterm is a Boolean function that is true at exactly one point.

For example, let's consider the more-or-less random function $f(w,x,y,z)$ whose truth table is

$$1100\,1001\,0000\,1111. \tag{22}$$

When this function is expanded by repeatedly applying Boole's law (20), we get a disjunction of eight minterms, one for each of the 1s in the truth table:

$$\begin{aligned} f(w,x,y,z) \;=\; &(\bar{w} \wedge \bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge \bar{x} \wedge \bar{y} \wedge z) \vee (\bar{w} \wedge x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{w} \wedge x \wedge y \wedge z) \\ &\vee (w \wedge x \wedge \bar{y} \wedge \bar{z}) \vee (w \wedge x \wedge \bar{y} \wedge z) \vee (w \wedge x \wedge y \wedge \bar{z}) \vee (w \wedge x \wedge y \wedge z). \end{aligned} \tag{23}$$

In general, a disjunction of minterms is called a *full disjunctive normal form.* Every Boolean function can be expressed in this way, and the result is unique—except, of course, for the order of the minterms. *Nitpick:* A special case arises when $f(x_1, \ldots, x_n)$ is identically zero. We consider '0' to be an empty disjunction, with no terms, and we also consider '1' to be an empty conjunction, for the same reasons as we defined $\sum_{k=1}^{0} a_k = 0$ and $\prod_{k=1}^{0} a_k = 1$ in Section 1.2.3.

C. S. Peirce observed, in *Amer. J. Math.* **3** (1880), 37–39, that every Boolean function also has a *full conjunctive normal form*, which is a conjunction of "minclauses" like $\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4 \vee x_5$. A minclause is 0 at only one point; so each clause in such a conjunction accounts for a place where the truth table has a 0. For example, the full conjunctive normal form of our function in (22) and (23) is

$$f(w,x,y,z) = (w\vee x\vee \bar{y}\vee z) \wedge (w\vee x\vee \bar{y}\vee \bar{z}) \wedge (w\vee \bar{x}\vee y\vee \bar{z}) \wedge (w\vee \bar{x}\vee \bar{y}\vee z) \\ \wedge (\bar{w}\vee x\vee y\vee z) \wedge (\bar{w}\vee x\vee y\vee \bar{z}) \wedge (\bar{w}\vee x\vee \bar{y}\vee z) \wedge (\bar{w}\vee x\vee \bar{y}\vee \bar{z}). \quad (24)$$

Not surprisingly, however, we often want to work with disjunctions and conjunctions that *don't* necessarily involve full minterms or minclauses. Therefore, following nomenclature introduced by Paul Bernays in his *Habilitationsschrift* (1918), we speak in general of a *disjunctive normal form* or "DNF" as *any* disjunction of conjunctions,

$$\bigvee_{j=1}^{m} \bigwedge_{k=1}^{s_j} u_{jk} = (u_{11} \wedge \cdots \wedge u_{1s_1}) \vee \cdots \vee (u_{m1} \wedge \cdots \wedge u_{ms_m}), \quad (25)$$

where each u_{jk} is a *literal*, namely a variable x_i or its complement. Similarly, any conjunction of disjunctions of literals,

$$\bigwedge_{j=1}^{m} \bigvee_{k=1}^{s_j} u_{jk} = (u_{11} \vee \cdots \vee u_{1s_1}) \wedge \cdots \wedge (u_{m1} \vee \cdots \vee u_{ms_m}), \quad (26)$$

is called a *conjunctive normal form*, or "CNF" for short.

A great many electrical circuits embedded inside today's computer chips are composed of "programmable logic arrays" (PLAs), which are ORs of ANDs of possibly complemented input signals. In other words, a PLA basically computes one or more disjunctive normal forms. Such building blocks are fast, versatile, and relatively inexpensive; and indeed, DNFs have played a prominent role in electrical engineering ever since the 1950s, when switching circuits were implemented with comparatively old-fashioned devices like relays or vacuum tubes. Therefore people have long been interested in finding the simplest DNFs for classes of Boolean functions, and we can expect that an understanding of disjunctive normal forms will continue to be important as technology continues to evolve.

The terms of a DNF are often called *implicants*, because the truth of any term in a disjunction implies the truth of the whole formula. In a formula like

$$f(x,y,z) = (x \wedge \bar{y} \wedge z) \vee (y \wedge z) \vee (\bar{x} \wedge y \wedge \bar{z}),$$

for example, we know that f is true when $x\wedge\bar{y}\wedge z$ is true, namely when $(x,y,z) = (1,0,1)$. But notice that in this example the shorter term $x \wedge z$ also turns out to

be an implicant of f, even though not written explicitly, because the additional term $y \wedge z$ makes the function true whenever $x = z = 1$, regardless of the value of y. Similarly, $\bar{x} \wedge y$ is an implicant of this particular function. So we might as well work with the simpler formula

$$f(x, y, z) = (x \wedge z) \vee (y \wedge z) \vee (\bar{x} \wedge y). \qquad (27)$$

At this point no more deletions are possible within the implicants, because neither x nor y nor z nor $\bar{x}$ is a strong enough condition to imply the truth of f.

An implicant that can't be factored further by removing any of its literals without making it too weak is called a *prime implicant*, following the terminology of W. V. Quine in *AMM* **59** (1952), 521–531.

These basic concepts can perhaps be understood most easily if we simplify the notation and adopt a more geometric viewpoint. We can write simply '$f(x)$' instead of $f(x_1, \ldots, x_n)$, and regard x as a vector, or as a binary string $x_1 \ldots x_n$ of length n. For example, the strings $wxyz$ where the function of (22) is true are

$$\{0000, 0001, 0100, 0111, 1100, 1101, 1110, 1111\}, \qquad (28)$$

and we can think of them as eight points in the 4-dimensional hypercube $2 \times 2 \times 2 \times 2$. The eight points in (28) correspond to the minterm implicants that are explicitly present in the full disjunctive normal form (23); but none of those implicants is actually prime. For example, the first two points of (28) make the subcube 000*, and the last four points constitute the subcube 11**, if we use asterisks to denote "wild cards" as we did when discussing database queries in Section 6.5; therefore $\bar{w} \wedge \bar{x} \wedge \bar{y}$ is an implicant of f, and so is $w \wedge x$. Similarly, we can see that the subcube 0*00 accounts for two of the eight points in (28), making $\bar{w} \wedge \bar{y} \wedge \bar{z}$ an implicant.

In general, each prime implicant corresponds in this way to a *maximal* subcube that stays within the set of points that make f true. (The subcube is maximal in the sense that it isn't contained in any larger subcube with the same property; we can't replace any of its explicit bits by an asterisk. A maximal subcube has a maximal number of asterisks, hence a minimal number of constrained coordinates, hence a minimal number of variables in the corresponding implicant.) The maximal subcubes of the eight points in (28) are

$$000*,\ 0*00,\ *100,\ *111,\ 11**; \qquad (29)$$

so the prime implicants of the function $f(w, x, y, z)$ in (23) are

$$(\bar{w} \wedge \bar{x} \wedge \bar{y}) \vee (\bar{w} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge z) \vee (w \wedge x). \qquad (30)$$

The *disjunctive prime form* of a Boolean function is the disjunction of all its prime implicants. Exercise 30 contains an algorithm to find all the prime implicants of a given function, based on a list of the points where the function is true.

We can define a *prime clause* in an exactly similar way: It is a disjunctive clause that is implied by f, having no subclause with the same property. And the *conjunctive prime form* of f is the conjunction of all its prime clauses. (An example appears in exercise 19.)

In many simple cases, the disjunctive prime form is the shortest possible disjunctive normal form that a function can have. But we can often do better, because we might be able to cover all the necessary points with only a few of the maximal subcubes. For example, the prime implicant $(y \wedge z)$ is unnecessary in (27). And in expression (30) we don't need both $(\bar{w} \wedge \bar{y} \wedge \bar{z})$ and $(x \wedge \bar{y} \wedge \bar{z})$; either one is sufficient, in the presence of the other terms.

Unfortunately, we will see in Section 7.9 that the task of finding a best disjunctive normal form is NP-complete, thus quite difficult in general. But many useful shortcuts have been developed for sufficiently small problems, and they are well explained in the book *Introduction to the Theory of Switching Circuits* by E. J. McCluskey (New York: McGraw–Hill, 1965). For later developments, see Petr Fišer and Jan Hlavička, *Computing and Informatics* **22** (2003), 19–51.

There's an important special case for which the shortest DNF is, however, easily characterized. A Boolean function is said to be *monotone* or *positive* if its value does not change from 1 to 0 when any of its variables changes from 0 to 1. In other words, f is monotone if and only if $f(x) \le f(y)$ whenever $x \subseteq y$, where the bit string $x = x_1 \ldots x_n$ is regarded as contained in or equal to the bit string $y = y_1 \ldots y_n$ if and only if $x_j \le y_j$ for all j. An equivalent condition (see exercise 21) is that the function f either is constant or can be expressed entirely in terms of $\wedge$ and $\vee$, without complementation.

Theorem Q. *The shortest disjunctive normal form of a monotone Boolean function is its disjunctive prime form.*

Proof. [W. V. Quine, *Boletín de la Sociedad Matemática Mexicana* **10** (1953), 64–70.] Let $f(x_1, \ldots, x_n)$ be monotone, and let $u_1 \wedge \cdots \wedge u_s$ be one of its prime implicants. We cannot have, say, $u_1 = \bar{x}_i$, because in that case the shorter term $u_2 \wedge \cdots \wedge u_s$ would also be an implicant, by monotonicity. Therefore no prime implicant has a complemented literal.

Now if we set $u_1 \leftarrow \cdots \leftarrow u_s \leftarrow 1$ and all other variables to 0, the value of f will be 1, but all of f's other prime implicants will vanish. Thus $u_1 \wedge \cdots \wedge u_s$ must be in every shortest DNF, because every implicant of a shortest DNF is clearly prime. ▮

Corollary Q. *A disjunctive normal form is the disjunctive prime form of a monotone Boolean function if and only if it has no complemented literals and none of its implicants is contained in another.* ▮

Satisfiability. A Boolean function is said to be *satisfiable* if it is not identically zero — that is, if it has at least one implicant. The most famous unsolved problem in all of computer science is to find an efficient way to decide whether a given Boolean function is satisfiable or unsatisfiable. More precisely, we ask: Is there an algorithm that inputs a Boolean formula of length N and tests it for satisfiability, always giving the correct answer after performing at most $N^{O(1)}$ steps?

When you hear about this problem for the first time, you might be tempted to ask a question of your own in return: "What? Are you serious that computer scientists still haven't figured out how to do such a simple thing?"

Well, if you think satisfiability testing is trivial, please tell us your method. We agree that the problem isn't always difficult; if, for example, the given formula involves only 30 Boolean variables, a brute-force trial of 2^{30} cases — that's about a billion — will indeed settle the matter. But an enormous number of practical problems that still await solution can be formulated as Boolean functions with, say, 100 variables, because mathematical logic is a very powerful way to express concepts. And the solutions to those problems correspond to the vectors $x = x_1 \ldots x_{100}$ for which $f(x) = 1$. So a truly efficient solution to the satisfiability problem would be a wonderful achievement.

There is at least one sense in which satisfiability testing is a no-brainer: If the function $f(x_1, \ldots, x_n)$ has been chosen at random, so that all 2^n truth tables are equally likely, then f is almost surely satisfiable, and we can find an x with $f(x) = 1$ after making fewer than 2 trials (on the average). It's like flipping a coin until it comes up heads; we rarely need to wait long. But the catch, of course, is that practical problems do not have random truth tables.

Okay, let's grant that satisfiability testing does seem to be tough, in general. In fact, satisfiability turns out to be difficult even when we try to simplify it by requiring that the Boolean function be presented as a "formula in 3CNF" — namely as a conjunctive normal form that has only *three* literals in each clause:

$$f(x_1, \ldots, x_n) = (t_1 \vee u_1 \vee v_1) \wedge (t_2 \vee u_2 \vee v_2) \wedge \cdots \wedge (t_m \vee u_m \vee v_m). \qquad (31)$$

Here each t_j, u_j, and v_j is x_k or $\bar{x}_k$ for some k. The problem of deciding satisfiability for formulas in 3CNF is called "3SAT," and exercise 39 explains why it is not really easier than satisfiability in general.

We will be seeing many examples of hard-to-crack 3SAT problems, especially in Section 7.9, where satisfiability testing will be discussed in great detail. The situation is a little peculiar, however, because a formula needs to be fairly long before we need to think twice about its satisfiability. For example, the shortest unsatisfiable formula in 3CNF is $(x \vee x \vee x) \wedge (\bar{x} \vee \bar{x} \vee \bar{x})$; but it is obviously no challenge to the intellect. We don't get into rough waters unless the three literals t_j, u_j, v_j of a clause correspond to three different variables. And in that case, each clause rules out exactly 1/8 of the possibilities, because seven different settings of (t_j, u_j, v_j) will make it true. Consequently every such 3CNF with at most seven clauses is automatically satisfiable, and a random setting of its variables will succeed with probability $\geq 1 - 7/8 = 1/8$.

The shortest interesting formula in 3CNF therefore has at least eight clauses. And in fact, an interesting 8-clause formula does exist, based on the associative block design by R. L. Rivest that we considered in 6.5–(13):

$$(x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$
$$\wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3). \qquad (32)$$

Any seven of these eight clauses are satisfiable, in exactly two ways, and they force the values of three variables; for example, the first seven imply that we have $x_1x_2x_3 = 001$. But the complete set of eight cannot be satisfied simultaneously.

Simple special cases. Two important classes of Boolean formulas have been identified for which the satisfiability problem does turn out to be pretty easy. These special cases arise when the conjunctive normal form being tested consists entirely of "Horn clauses" or entirely of "Krom clauses." A *Horn clause* is an OR of literals in which all or nearly all of the literals are complemented — at most one of its literals is a pure, unbarred variable. A *Krom clause* is an OR of exactly two literals. Thus, for example,

$$\bar{x} \vee \bar{y}, \qquad w \vee \bar{y} \vee \bar{z}, \qquad \bar{u} \vee \bar{v} \vee \bar{w} \vee \bar{x} \vee \bar{y} \vee z, \quad \text{and} \quad x$$

are examples of Horn clauses; and

$$x \vee x, \qquad \bar{x} \vee \bar{x}, \qquad \bar{x} \vee \bar{y}, \qquad x \vee \bar{y}, \qquad \bar{x} \vee y, \quad \text{and} \quad x \vee y$$

are examples of Krom clauses, only the last of which is not also a Horn clause. (The first example qualifies because $x \vee x = x$.) Notice that a Horn clause is allowed to contain any number of literals, but when we restrict ourselves to Krom clauses we are essentially considering the 2SAT problem. In both cases we will see that satisfiability can be decided in linear time — that is, by carrying out only $O(N)$ simple steps, when given a formula of length N.

Let's consider Horn clauses first. Why are they so easy to handle? The main reason is that a clause like $\bar{u} \vee \bar{v} \vee \bar{w} \vee \bar{x} \vee \bar{y} \vee z$ can be recast in the form $\neg(u \wedge v \wedge w \wedge x \wedge y) \vee z$, which is the same as

$$u \wedge v \wedge w \wedge x \wedge y \;\Rightarrow\; z.$$

In other words, if u, v, w, x, and y are all true, then z must also be true. For this reason, parameterized Horn clauses were chosen to be the basic underlying mechanism of the programming language called Prolog. Furthermore there is an easy way to characterize exactly which Boolean functions can be represented entirely with Horn clauses:

Theorem H. *The Boolean function $f(x_1, \ldots, x_n)$ is expressible as a conjunction of Horn clauses if and only if*

$$f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) = 1 \quad \text{implies} \quad f(x_1 \wedge y_1, \ldots, x_n \wedge y_n) = 1 \quad (33)$$

for all Boolean values x_j and y_j.

Proof. [Alfred Horn, *J. Symbolic Logic* **16** (1951), 14–21, Lemma 7.] If we have $x_0 \vee \bar{x}_1 \vee \cdots \vee \bar{x}_k = 1$ and $y_0 \vee \bar{y}_1 \vee \cdots \vee \bar{y}_k = 1$, then

$$\begin{aligned}(x_0 \wedge y_0) &\vee \overline{x_1 \wedge y_1} \vee \cdots \vee \overline{x_k \wedge y_k} \\ &= (x_0 \vee \bar{x}_1 \vee \bar{y}_1 \vee \cdots \vee \bar{x}_k \vee \bar{y}_k) \wedge (y_0 \vee \bar{x}_1 \vee \bar{y}_1 \vee \cdots \vee \bar{x}_k \vee \bar{y}_k) \\ &\geq (x_0 \vee \bar{x}_1 \vee \cdots \vee \bar{x}_k) \wedge (y_0 \vee \bar{y}_1 \vee \cdots \vee \bar{y}_k) \;=\; 1;\end{aligned}$$

and a similar (but simpler) calculation applies when the unbarred literals x_0 and y_0 are not present. Therefore every conjunction of Horn clauses satisfies (33).

Conversely, condition (33) implies that every prime clause of f is a Horn clause (see exercise 44). ∎

Let's say that a *Horn function* is a Boolean function that satisfies condition (33), and let's also call it *definite* if it satisfies the further condition $f(1,\ldots,1)=1$. It's easy to see that a conjunction of Horn clauses is definite if and only if each clause has *exactly* one unbarred literal, because only an entirely negative clause like $\bar{x}\vee\bar{y}$ will fail if all variables are true. Definite Horn functions are slightly simpler to work with than Horn functions in general, because they are obviously always satisfiable. Thus, by Theorem H, they have a unique least vector x such that $f(x)=1$, namely the bitwise AND of all vectors that satisfy all clauses. The *core* of a definite Horn function is the set of all variables x_j that are true in this minimum vector x. Notice that the variables in the core must be true whenever f is true, so we can essentially factor them out.

Definite Horn functions arise in many ways, for example in the analysis of games (see exercises 51 and 52). Another nice example comes from compiler technology. Consider the following typical (but simplified) grammar for algebraic expressions in a programming language:

⟨expression⟩ → ⟨term⟩ | ⟨expression⟩ + ⟨term⟩ | ⟨expression⟩ - ⟨term⟩
⟨term⟩ → ⟨factor⟩ | -⟨factor⟩ | ⟨term⟩ * ⟨factor⟩ | ⟨term⟩ / ⟨factor⟩
⟨factor⟩ → ⟨variable⟩ | ⟨constant⟩ | (⟨expression⟩)
⟨variable⟩ → ⟨letter⟩ | ⟨variable⟩⟨letter⟩ | ⟨variable⟩⟨digit⟩   (34)
⟨letter⟩ → a | b | c
⟨constant⟩ → ⟨digit⟩ | ⟨constant⟩⟨digit⟩
⟨digit⟩ → 0 | 1

For example, the string `a/(-b0-10)+cc*cc` meets the syntax for ⟨expression⟩ and uses each of the grammatical rules at least once.

Suppose we want to know what pairs of characters can appear next to each other in such expressions. Definite Horn clauses provide the answer, because we can set the problem up as follows: Let the quantities **Xx**, **xX**, and **xy** denote Boolean "propositions," where X is one of the symbols {E, T, F, V, L, C, D} standing respectively for ⟨expression⟩, ⟨term⟩, ..., ⟨digit⟩, and where x and y are symbols in the set {+, -, *, /, (,), a, b, c, 0, 1}. The proposition **Xx** means, "X can end with x"; similarly, **xX** means, "X can start with x"; and **xy** means, "The character x can be followed immediately by y in an expression." (There are $7\times 11+11\times 7+11\times 11=275$ propositions altogether.) Then we can write

xT ⇒ xE	⇒ -T	xC ⇒ xF	Vx ∧ yL ⇒ xy	⇒ Lc	
Tx ⇒ Ex	xF ⇒ -x	Cx ⇒ Fx	Vx ∧ yD ⇒ xy	xD ⇒ xC	
Ex ⇒ x+	Tx ⇒ x*	⇒ (F	Dx ⇒ Vx	Dx ⇒ Cx	
xT ⇒ +x	xF ⇒ *x	xE ⇒ (x	⇒ aL	Cx ∧ yD ⇒ xy	(35)
Ex ⇒ x-	Tx ⇒ x/	Ex ⇒ x)	⇒ La	⇒ 0D	
xT ⇒ -x	xF ⇒ /x	⇒ F)	⇒ bL	⇒ D0	
xF ⇒ xT	xV ⇒ xF	xL ⇒ xV	⇒ Lb	⇒ 1D	
Fx ⇒ Tx	Vx ⇒ Fx	Lx ⇒ Vx	⇒ cL	⇒ D1	

where x and y run through the eleven terminal symbols {+, ..., 1}. This schematic specification gives us a total of $24\times 11+3\times 11\times 11+13\times 1=640$ definite

Horn clauses, which we could write out formally as

$$(\overline{\texttt{+T}} \vee \texttt{+E}) \wedge (\overline{\texttt{-T}} \vee \texttt{-E}) \wedge \cdots \wedge (\overline{\texttt{V+}} \vee \overline{\texttt{0L}} \vee \texttt{+0}) \wedge \cdots \wedge (\texttt{D1})$$

if we prefer the cryptic notation of Boolean algebra to the $\Rightarrow$ convention of (35).

Why did we do this? Because *the core of all these clauses is the set of all propositions that are true in this particular grammar.* For example, one can verify that -E is true, hence the symbols (- can occur next to each other within an expression; but the symbol pairs ++ and *- cannot (see exercise 46).

Furthermore, we can find the core of any given set of definite Horn clauses without great difficulty. We just start out with the propositions that appear alone, on the right-hand side of $\Rightarrow$ when the left-hand side is empty; thirteen clauses of that kind appear in (35). And once we assert the truth of those propositions, we might find one or more clauses whose left-hand sides are now known to be true. Hence their right-hand sides also belong to the core, and we can keep going in the same way. The whole procedure is pretty much like letting water run downhill until it has found its proper level. In fact, when we choose appropriate data structures, this downhill process goes quite fast, requiring only $O(N+n)$ steps, when N denotes the total length of the clauses and n is the number of propositional variables. (We assume here that all clauses have been expanded out, not abbreviated in terms of parameters like x and y above. More sophisticated techniques of theorem proving are available to deal with parameterized clauses, but they are beyond the scope of our present discussion.)

Algorithm C (*Core computation for definite Horn clauses*). Given a set P of propositional variables and a set C of clauses, each having the form

$$u_1 \wedge \cdots \wedge u_k \;\Rightarrow\; v \qquad \text{where } k \geq 0 \text{ and } \{u_1, \ldots, u_k, v\} \subseteq P, \qquad (36)$$

this algorithm finds the set $Q \subseteq P$ of all propositional variables that are necessarily true whenever all of the clauses are true.

We use the following data structures for clauses c, propositions p, and hypotheses h, where a "hypothesis" is the appearance of a proposition on the left-hand side of a clause:

CONCLUSION(c) is the proposition on the right of clause c;

COUNT(c) is the number of hypotheses of c not yet asserted;

TRUTH(p) is 1 if p is known to be true, otherwise 0;

LAST(p) is the last hypothesis in which p appears;

CLAUSE(h) is the clause for which h appears on the left;

PREV(h) is the previous hypothesis containing the proposition of h.

We also maintain a stack S_0, S_1, ..., S_{s-1} of all propositions that are known to be true but not yet asserted.

C1. [Initialize.] Set LAST(p) $\leftarrow \Lambda$ and TRUTH(p) $\leftarrow 0$ for each proposition p. Also set $s \leftarrow 0$, so that the stack is empty. Then for each clause c, having the form (36), set CONCLUSION(c) $\leftarrow v$ and COUNT(c) $\leftarrow k$. If $k = 0$ and

TRUTH(v) $= 0$, set TRUTH(v) $\leftarrow 1$, $S_s \leftarrow v$, and $s \leftarrow s+1$. Otherwise, for $1 \le j \le k$, create a hypothesis record h and set CLAUSE(h) $\leftarrow c$, PREV(h) $\leftarrow$ LAST(u_j), LAST(u_j) $\leftarrow h$.

C2. [Prepare to assert p.] Terminate the algorithm if $s = 0$; the desired core now consists of all propositions whose TRUTH has been set to 1. Otherwise set $s \leftarrow s - 1$, $p \leftarrow S_s$, and $h \leftarrow$ LAST(p).

C3. [Done with hypotheses?] If $h = \Lambda$, return to C2.

C4. [Validate h.] Set $c \leftarrow$ CLAUSE(h) and COUNT(c) $\leftarrow$ COUNT(c) $- 1$. If the new value of COUNT(c) is still nonzero, go to step C6.

C5. [Deduce CONCLUSION(c).] Set $p \leftarrow$ CONCLUSION(c). If TRUTH(p) $= 0$, set TRUTH(p) $\leftarrow 1$, $S_s \leftarrow p$, $s \leftarrow s+1$.

C6. [Loop on h.] Set $h \leftarrow$ PREV(h) and return to C3. ▮

Notice how smoothly the data structures work together, avoiding any need to search for a place to make progress in the calculation. Algorithm C is similar in many respects to Algorithm 2.2.3T (topological sorting), which was the first example of multilinked data structures that we discussed long ago in Chapter 2; in fact, we can regard Algorithm 2.2.3T as the special case of Algorithm C in which every proposition appears on the right-hand side of exactly one clause. (See exercise 47.)

Exercise 48 shows that a slight modification of Algorithm C solves the satisfiability problem for Horn clauses in general. Further discussion can be found in a paper by W. F. Dowling and J. H. Gallier, *J. Logic Programming* **1** (1984), 267–284.

We turn now to Krom functions and the 2SAT problem. Again there's a linear-time algorithm; but again, we can probably appreciate it best if we look first at a simplified-but-practical application. Let's suppose that seven comedians have each agreed to do one-night standup gigs at two of five hotels during a three-day festival, but each of them is available for only two of those days because of other commitments:

Tomlin should do Aladdin and Caesars on days 1 and 2;
Unwin should do Bellagio and Excalibur on days 1 and 2;
Vegas should do Desert and Excalibur on days 2 and 3;
Williams should do Aladdin and Desert on days 1 and 3; (37)
Xie should do Caesars and Excalibur on days 1 and 3;
Yankovic should do Bellagio and Desert on days 2 and 3;
Zany should do Bellagio and Caesars on days 1 and 2.

Is it possible to schedule them all without conflict?

To solve this problem, we can introduce seven Boolean variables $\{t, u, v, w, x, y, z\}$, where t (for example) means that Tomlin does Aladdin on day 1 and Caesars on day 2 while $\bar{t}$ means that the days booked for those hotels occur in the opposite order. Then we can set up constraints to ensure that no two comedians

are booked in the same hotel on the same day:

$$\begin{array}{llll} \neg(t\wedge w)\ \text{[A1]} & \neg(y\wedge \bar z)\ \text{[B2]} & \neg(t\wedge z)\ \text{[C2]} & \neg(w\wedge y)\ \text{[D3]} \\ \neg(u\wedge z)\ \text{[B1]} & \neg(\bar t\wedge x)\ \text{[C1]} & \neg(v\wedge \bar y)\ \text{[D2]} & \neg(\bar u\wedge \bar x)\ \text{[E1]} \\ \neg(\bar u\wedge y)\ \text{[B2]} & \neg(\bar t\wedge \bar z)\ \text{[C1]} & \neg(\bar v\wedge w)\ \text{[D3]} & \neg(u\wedge \bar v)\ \text{[E2]} \\ \neg(\bar u\wedge \bar z)\ \text{[B2]} & \neg(x\wedge \bar z)\ \text{[C1]} & \neg(\bar v\wedge y)\ \text{[D3]} & \neg(v\wedge x)\ \text{[E3]} \end{array} \qquad (38)$$

Each of these constraints is, of course, a Krom clause; we must satisfy

$$(\bar t\vee\bar w)\wedge(\bar u\vee\bar z)\wedge(u\vee\bar y)\wedge(u\vee z)\wedge(\bar y\vee z)\wedge(t\vee\bar x)\wedge(t\vee z)\wedge(\bar x\vee z)$$
$$\wedge(\bar t\vee\bar z)\wedge(\bar v\vee y)\wedge(v\vee\bar w)\wedge(v\vee\bar y)\wedge(\bar w\vee\bar y)\wedge(u\vee x)\wedge(\bar u\vee v)\wedge(\bar v\vee\bar x). \qquad (39)$$

Furthermore, Krom clauses (like Horn clauses) can be written as implications:

$$t\Rightarrow\bar w,\quad u\Rightarrow\bar z,\quad \bar u\Rightarrow\bar y,\quad \bar u\Rightarrow z,\quad y\Rightarrow z,\quad \bar t\Rightarrow\bar x,\quad \bar t\Rightarrow z,\quad x\Rightarrow z,$$
$$t\Rightarrow\bar z,\quad v\Rightarrow y,\quad \bar v\Rightarrow\bar w,\quad \bar v\Rightarrow\bar y,\quad w\Rightarrow\bar y,\quad \bar u\Rightarrow x,\quad u\Rightarrow v,\quad v\Rightarrow\bar x. \qquad (40)$$

And every such implication also has an alternative, "contrapositive" form:

$$w\Rightarrow\bar t,\quad z\Rightarrow\bar u,\quad y\Rightarrow u,\quad \bar z\Rightarrow u,\quad \bar z\Rightarrow\bar y,\quad x\Rightarrow t,\quad \bar z\Rightarrow t,\quad \bar z\Rightarrow\bar x,$$
$$z\Rightarrow\bar t,\quad \bar y\Rightarrow\bar v,\quad w\Rightarrow v,\quad y\Rightarrow v,\quad y\Rightarrow\bar w,\quad \bar x\Rightarrow u,\quad \bar v\Rightarrow\bar u,\quad x\Rightarrow\bar v. \qquad (41)$$

But oops — alas — there is a vicious cycle,

$$u \underset{\text{[B1]}}{\Rightarrow} \bar z \underset{\text{[B2]}}{\Rightarrow} \bar y \underset{\text{[D2]}}{\Rightarrow} \bar v \underset{\text{[E2]}}{\Rightarrow} \bar u \underset{\text{[B2]}}{\Rightarrow} z \underset{\text{[C2]}}{\Rightarrow} \bar t \underset{\text{[C1]}}{\Rightarrow} \bar x \underset{\text{[E1]}}{\Rightarrow} u \qquad (42)$$

This cycle tells that u and $\bar u$ must both have the same value; so there is no way to accommodate all of the conditions in (37). The festival organizers will have to renegotiate their agreement with at least one of the six comedians $\{t, u, v, x, y, z\}$, if a viable schedule is to be achieved. (See exercise 53.)

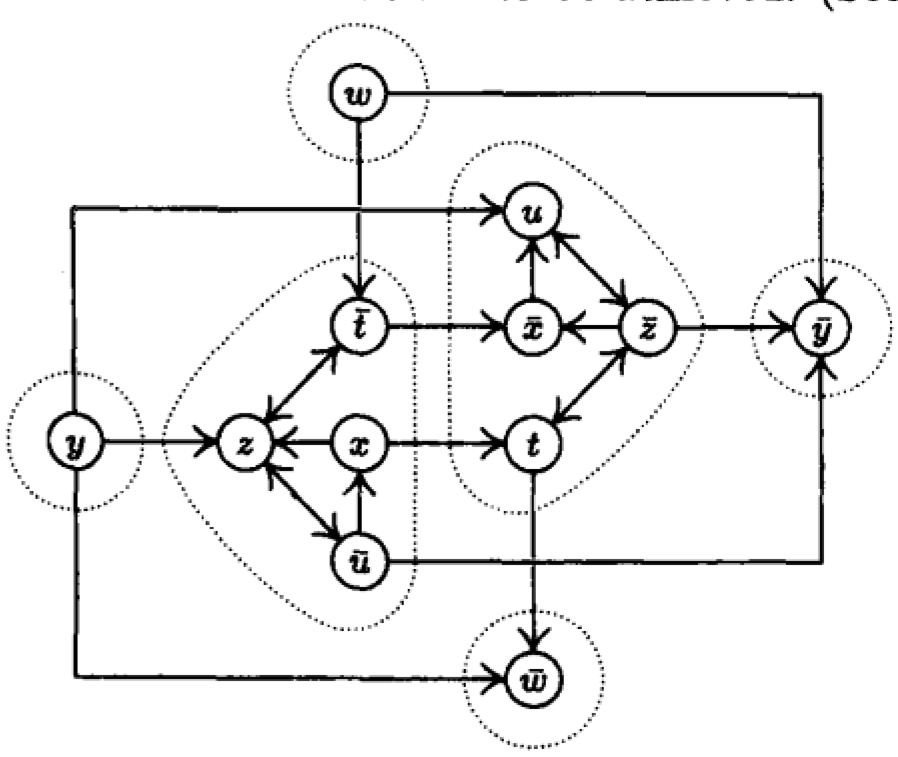


Fig. 6. The digraph corresponding to all implications of (40) and (41) that do not involve either v or $\bar v$. Assigning appropriate values to the literals in each strong component will solve a binary scheduling problem that is an instance of 2SAT.

The organizers might, for instance, try to leave v out of the picture temporarily. Then five of the sixteen constraints in (38) would go away and only 22 of the implications from (40) and (41) would remain, leaving the directed graph illustrated in Fig. 6. This digraph does contain cycles, like $z \Rightarrow \bar u \Rightarrow x \Rightarrow z$ and $t \Rightarrow \bar z \Rightarrow t$; but no cycle contains both a variable and its complement. Indeed,

we can see from Fig. 6 that the values $tuwxyz = 110000$ do satisfy every clause of (39) that doesn't involve v or $\bar{v}$. These values give us a schedule that satisfies six of the seven original stipulations in (37), starting with (Tomlin, Unwin, Zany, Williams, Xie) at the (Aladdin, Bellagio, Caesars, Desert, Excalibur) on day 1.

In general, given any 2SAT problem with m Krom clauses that involve n Boolean variables, we can form a directed graph in the same way. There are $2n$ vertices $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$, one for each possible literal; and there are $2m$ arcs of the form $\bar{u} \to v$ and $\bar{v} \to u$, two for each clause $u \vee v$. Two literals u and v belong to the same *strong component* of this digraph if and only if there are oriented paths from u to v and from v to u. For example, the six strong components of the digraph in Fig. 6 are indicated by dotted contours. All literals in a strong component must have the same Boolean value, in any solution to the corresponding 2SAT problem.

Theorem K. *A conjunctive normal form with two literals per clause is satisfiable if and only if no strong component of the associated digraph contains both a variable and its complement.*

Proof. [Melven Krom, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **13** (1967), 15–20, Corollary 2.2.] If there are paths from x to $\bar{x}$ and from $\bar{x}$ to x, the formula is certainly unsatisfiable.

Conversely, assume that no such paths exist. Any digraph has at least one strong component S that is a "source," having no incoming arcs from vertices in any other strong component. Moreover, our digraph always has an attractive antisymmetry, illustrated in Fig. 6: We have $u \to v$ if and only if $\bar{v} \to \bar{u}$. Therefore the complements of the literals in S form another strong component $\overline{S} \neq S$ that is a "sink," having no *outgoing* arcs to other strong components. Hence we can assign the value 0 to all literals in S and 1 to all literals in $\overline{S}$, then remove them from the digraph and proceed in the same way until all literals have received a value. The resulting values satisfy $u \leq v$ whenever $u \to v$ in the digraph; hence they satisfy $\bar{u} \vee v$ whenever $\bar{u} \vee v$ is a clause of the formula. ∎

Theorem K leads immediately to an efficient solution of the 2SAT problem, thanks to an algorithm by R. E. Tarjan that finds strong components in linear time. [See *SICOMP* **1** (1972), 146–160; D. E. Knuth, *The Stanford GraphBase* (1993), 512–519.] We shall study Tarjan's algorithm in detail in Section 7.4.1. Exercise 54 shows that the condition of Theorem K is readily checked whenever the algorithm detects a new strong component. Furthermore, the algorithm detects "sinks" first; thus, as a simple byproduct of Tarjan's procedure, we can assign values that establish satisfiability by choosing the value 1 for each literal in a strong component that occurs before its complement.

Medians. We've been focusing on Boolean binary operations like $x \vee y$ or $x \oplus y$. But there's also a significant *ternary* operation $\langle xyz \rangle$, called the *median* of x, y, and z:

$$\langle xyz \rangle \;=\; (x \wedge y) \vee (y \wedge z) \vee (x \wedge z) = (x \vee y) \wedge (y \vee z) \wedge (x \vee z). \qquad (43)$$

In fact, $\langle xyz\rangle$ is probably the most important ternary operation in the entire universe, because it has amazing properties that are continually being discovered and rediscovered.

In the first place, we can see easily that this formula for $\langle xyz\rangle$ describes the *majority* value of any three Boolean quantities x, y, and z: $\langle 000\rangle = \langle 001\rangle = 0$ and $\langle 011\rangle = \langle 111\rangle = 1$. We call $\langle xyz\rangle$ the "median" instead of the "majority" because, if x, y, and z are arbitrary *real* numbers, and if the operations $\wedge$ and $\vee$ denote min and max, then

$$\langle xyz\rangle \;=\; y \qquad \text{when } x \le y \le z. \tag{44}$$

Secondly, the basic binary operations $\wedge$ and $\vee$ are special cases of medians:

$$x \wedge y \;=\; \langle x0y\rangle; \qquad\qquad x \vee y \;=\; \langle x1y\rangle. \tag{45}$$

Thus *any* monotone Boolean function can be expressed entirely in terms of the ternary median operator and the constants 0 and 1. In fact, if we lived in a median-only world, we could let $\wedge$ stand for falsehood and $\vee$ for truth; then $x \wedge y = \langle x{\wedge}y\rangle$ and $x \vee y = \langle x{\vee}y\rangle$ would be perfectly natural expressions, and we could even use Polish notation like $\langle{\wedge}xy\rangle$ and $\langle{\vee}xy\rangle$ if that was our preference! The same idea applies to extended real numbers under the min-max interpretation of $\wedge$ and $\vee$, if we take medians with respect to the constants $\wedge = -\infty$ and $\vee = +\infty$.

A Boolean function $f(x_1, x_2, \ldots, x_n)$ is called *self-dual* when it satisfies

$$\overline{f(x_1, x_2, \ldots, x_n)} \;=\; f(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n). \tag{46}$$

We've noted that a Boolean function is monotone if and only if it can be expressed in terms of $\wedge$ and $\vee$; by De Morgan's laws (11) and (12), a monotone formula is self-dual if and only if the symbols $\wedge$ and $\vee$ can be interchanged without changing the formula's value. Thus the median operation defined in (43) is both monotone and self-dual. In fact, it is the simplest nontrivial function of that kind, since none of the binary operations in Table 1 are both monotone and self-dual except the projections ⊑ and ⊐.

Furthermore, *any* expression that has been formed entirely with the median operator, without using constants, is both monotone and self-dual. For example, the function $\langle w\langle xyz\rangle\langle w\langle uvw\rangle x\rangle\rangle$ is self-dual because

$$\begin{aligned}\overline{\langle w\langle xyz\rangle\langle w\langle uvw\rangle x\rangle\rangle} \;&=\; \langle \bar{w}\,\overline{\langle xyz\rangle}\,\overline{\langle w\langle uvw\rangle x\rangle}\rangle\\ &=\; \langle \bar{w}\langle \bar{x}\bar{y}\bar{z}\rangle\langle \bar{w}\overline{\langle uvw\rangle}\bar{x}\rangle\rangle \;=\; \langle \bar{w}\langle \bar{x}\bar{y}\bar{z}\rangle\langle \bar{w}\langle \bar{u}\bar{v}\bar{w}\rangle\bar{x}\rangle\rangle.\end{aligned}$$

Emil Post, while working on his Ph.D. thesis (Columbia University, 1920), proved that the converse statement is also true:

Theorem P. *Every monotone, self-dual Boolean function $f(x_1, \ldots, x_n)$ can be expressed entirely in terms of the median operation $\langle xyz\rangle$.*

Proof. [*Annals of Mathematics Studies* **5** (1941), 74–75.] Observe first that

$$\begin{aligned}&\langle x_1 y\langle x_2 y \ldots y\langle x_{s-1} y x_s\rangle \ldots\rangle\rangle\\ &\quad= \bigl((x_1 \vee x_2 \vee \cdots \vee x_{s-1} \vee x_s) \wedge y\bigr) \vee (x_1 \wedge x_2 \wedge \cdots \wedge x_{s-1} \wedge x_s); \end{aligned} \tag{47}$$

this formula for repeated medianing is easily proved by induction on s.

Now suppose $f(x_1, \ldots, x_n)$ is monotone, self-dual, and has the disjunctive prime form

$$f(x_1,\ldots,x_n) \;=\; t_1 \vee \cdots \vee t_m, \qquad t_j = x_{j1} \wedge \cdots \wedge x_{js_j},$$

where no prime implicant t_j is contained in another (Corollary Q). Any two prime implicants must have at least one variable in common. For if we had, say, $t_1 = x \wedge y$ and $t_2 = u \wedge v \wedge w$, the value of f would be 1 when $x = y = 1$ and $u = v = w = 0$, as well as when $x = y = 0$ and $u = v = w = 1$, contradicting self-duality. Therefore if any t_j consists of a single variable x, it must be the only prime implicant — in which case f is the trivial function $f(x_1, \ldots, x_n) = x = \langle xxx\rangle$.

Define the functions g_0, g_1, ..., g_m by composing medians as follows:

$$\begin{aligned} &g_0(x_1,\ldots,x_n) = x_1;\\ &g_j(x_1,\ldots,x_n) = h(x_{j1},\ldots,x_{js_j}; g_{j-1}(x_1,\ldots,x_n)), \text{ for } 1 \le j \le m; \end{aligned} \tag{48}$$

here $h(x_1, \ldots, x_s; y)$ denotes the function on the top line of (47). By induction on j, we can prove from (47) and (48) that $g_j(x_1, \ldots, x_n) = 1$ whenever we have $t_1 \vee \cdots \vee t_j = 1$, because $(x_{j1} \vee \cdots \vee x_{js_j}) \wedge t_k = t_k$ when $k < j$.

Finally, $f(x_1, \ldots, x_n)$ must equal $g_m(x_1, \ldots, x_n)$, because both functions are monotone and self-dual, and we have shown that $f(x_1, \ldots, x_n) \le g_m(x_1, \ldots, x_n)$ for all combinations of 0s and 1s. This inequality suffices to prove equality, because a self-dual function equals 1 in exactly half of the 2^n possible cases. ▌

One consequence of Theorem P is that we can express the median of five elements via medians of three, because the median of any odd number of Boolean variables is obviously a monotone and self-dual Boolean function. Let's write $\langle x_1 \ldots x_{2k-1}\rangle$ for such a median. Then the disjunctive prime form of $\langle vwxyz\rangle$ is

$$\begin{aligned}&(v\wedge w\wedge x) \vee (v\wedge w\wedge y) \vee (v\wedge w\wedge z) \vee (v\wedge x\wedge y) \vee (v\wedge x\wedge z)\\ &\qquad \vee (v\wedge y\wedge z) \vee (w\wedge x\wedge y) \vee (w\wedge x\wedge z) \vee (w\wedge y\wedge z) \vee (x\wedge y\wedge z);\end{aligned}$$

so the construction in the proof of Theorem P expresses $\langle vwxyz\rangle$ as a huge formula $g_{10}(v, w, x, y, z)$ involving 2,046 median-of-3 operations. Of course this expression isn't the shortest possible one; we actually have

$$\langle vwxyz\rangle \;=\; \bigl\langle v\langle xyz\rangle\langle wx\langle wyz\rangle\rangle\bigr\rangle. \tag{49}$$

[See H. S. Miiller and R. O. Winder, *IRE Transactions* **EC-11** (1962), 89–90.]

***Median algebras and median graphs.** We noted earlier that the ternary operation $\langle xyz\rangle$ is useful when x, y, and z belong to any ordered set like the real numbers, when $\wedge$ and $\vee$ are regarded as the operators min and max. In fact, the operation $\langle xyz\rangle$ also plays a useful role in far more general circumstances.

A *median algebra* is any set M on which a ternary operation $\langle xyz\rangle$ is defined that takes elements of M into elements of M and obeys the following three axioms:

$$\langle xxy\rangle = x \quad \text{(majority law);} \tag{50}$$

$$\langle xyz\rangle = \langle xzy\rangle = \langle yxz\rangle = \langle yzx\rangle = \langle zxy\rangle = \langle zyx\rangle \quad \text{(commutative law);} \tag{51}$$

$$\langle xw\langle ywz\rangle\rangle = \langle\langle xwy\rangle wz\rangle \quad \text{(associative law).} \tag{52}$$

In the Boolean case, for example, the associative law (52) holds for $w = 0$ and $w = 1$ because $\wedge$ and $\vee$ are associative. Exercises 75 and 76 prove that these three axioms imply also a *distributive law* for medians, which has both a short form

$$\big\langle\langle xyz\rangle uv\big\rangle \;=\; \big\langle x\langle yuv\rangle\langle zuv\rangle\big\rangle \tag{53}$$

and a more symmetrical long form

$$\big\langle\langle xyz\rangle uv\big\rangle \;=\; \big\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\big\rangle. \tag{54}$$

No simple proof of this fact is known, but we can at least verify the special case of (53) and (54) when $y = u$ and $z = v$: We have

$$\big\langle\langle xyz\rangle yz\big\rangle \;=\; \langle xyz\rangle \tag{55}$$

because both sides equal $\langle xy\langle zyz\rangle\rangle$. In fact, the associative law (52) is just the special case $y = u$ of (53). And with (55) and (52) we can also verify the case $x = u$: $\big\langle\langle uyz\rangle uv\big\rangle = \big\langle vu\langle yuz\rangle\big\rangle = \big\langle\langle vuy\rangle uz\big\rangle = \big\langle\langle yuv\rangle uz\big\rangle = \big\langle\langle\langle yuv\rangle uv\rangle uz\big\rangle = \big\langle\langle yuv\rangle u\langle vuz\rangle\big\rangle = \big\langle u\langle yuv\rangle\langle zuv\rangle\big\rangle$.

An *ideal* in a median algebra M is a set $C \subseteq M$ for which we have

$$\langle xyz\rangle \in C \qquad \text{whenever } x \in C,\ y \in C, \text{ and } z \in M. \tag{56}$$

If u and v are any elements of M, the *interval* $[u\,..\,v]$ is defined as follows:

$$[u\,..\,v] \;=\; \big\{\langle xuv\rangle \;\big|\; x \in M\big\}. \tag{57}$$

We say that "x is between u and v" if and only if $x \in [u\,..\,v]$. According to these definitions, u and v themselves always belong to the interval $[u\,..\,v]$.

Lemma M. *Every interval* $[u\,..\,v]$ *is an ideal, and* $x \in [u\,..\,v] \iff x = \langle uxv\rangle$.

Proof. Let $\langle xuv\rangle$ and $\langle yuv\rangle$ be arbitrary elements of $[u\,..\,v]$. Then

$$\big\langle\langle xuv\rangle\langle yuv\rangle z\big\rangle \;=\; \big\langle\langle xyz\rangle uv\big\rangle \;\in\; [u\,..\,v]$$

for all $z \in M$, by (51) and (53), so $[u\,..\,v]$ is an ideal. Furthermore every element $\langle xuv\rangle \in [u\,..\,v]$ satisfies $\langle xuv\rangle = \big\langle u\langle xuv\rangle v\big\rangle$ by (51) and (55). ∎

Our intervals $[u\,..\,v]$ have nice properties, because of the median laws:

$$v \in [u\,..\,u] \implies u = v; \tag{58}$$

$$x \in [u\,..\,v] \text{ and } y \in [u\,..\,x] \implies y \in [u\,..\,v]; \tag{59}$$

$$x \in [u\,..\,v] \text{ and } y \in [u\,..\,z] \text{ and } y \in [v\,..\,z] \implies y \in [x\,..\,z]. \tag{60}$$

Equivalently, $[u\,..\,u] = \{u\}$; if $x \in [u\,..\,v]$ then $[u\,..\,x] \subseteq [u\,..\,v]$; and $x \in [u\,..\,v]$ also implies that $[u\,..\,z] \cap [v\,..\,z] \subseteq [x\,..\,z]$ for all z. (See exercise 72.)

Now let's define a graph on the vertex set M, with the following edges:

$$u \text{---} v \iff u \neq v \text{ and } \langle xuv \rangle \in \{u, v\} \text{ for all } x \in M. \qquad (61)$$

In other words, u and v are adjacent if and only if the interval $[u \mathinner{.\,.} v]$ consists of just the two points u and v.

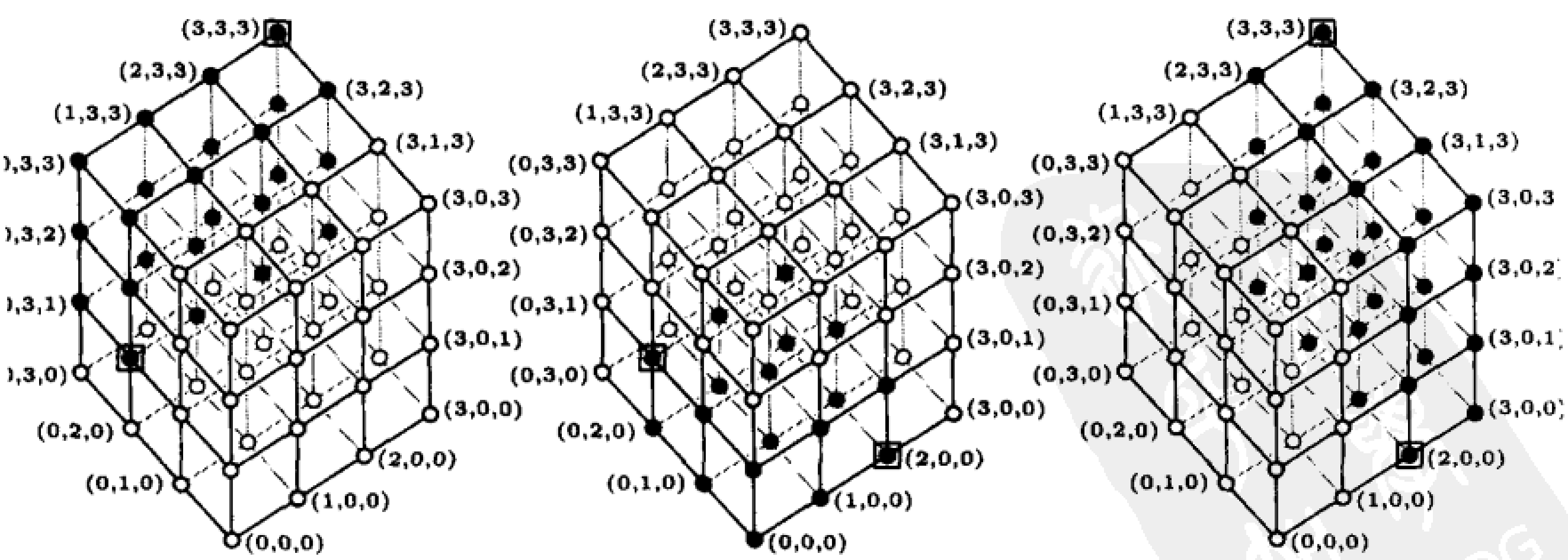
Theorem G. *If M is any finite median algebra, the graph defined by* (61) *is connected. Moreover, vertex x belongs to the interval $[u \mathinner{.\,.} v]$ if and only if x lies on a shortest path from u to v.*

Proof. If M isn't connected, choose u and v so that there is no path from u to v and the interval $[u \mathinner{.\,.} v]$ has as few elements as possible. Let $x \in [u \mathinner{.\,.} v]$ be distinct from u and v. Then $\langle xuv \rangle = x \neq v$, so $v \notin [u \mathinner{.\,.} x]$; similarly $u \notin [x \mathinner{.\,.} v]$. But $[u \mathinner{.\,.} x]$ and $[x \mathinner{.\,.} v]$ are contained in $[u \mathinner{.\,.} v]$, by (59). So they are smaller intervals, and there must be a path from u to x and from x to v. Contradiction.

The other half of the theorem is proved in exercise 73. ▮

Our definition of intervals implies that $\langle xyz \rangle \in [x \mathinner{.\,.} y] \cap [x \mathinner{.\,.} z] \cap [y \mathinner{.\,.} z]$, because $\langle xyz \rangle = \langle \langle xyz \rangle xy \rangle = \langle \langle xyz \rangle xz \rangle = \langle \langle xyz \rangle yz \rangle$ by (55). Conversely, if $w \in [x \mathinner{.\,.} y] \cap [x \mathinner{.\,.} z] \cap [y \mathinner{.\,.} z]$, exercise 74 proves that $w = \langle xyz \rangle$. In other words, *the intersection $[x \mathinner{.\,.} y] \cap [x \mathinner{.\,.} z] \cap [y \mathinner{.\,.} z]$ always contains exactly one point,* whenever x, y, and z are points of M.

Figure 7 illustrates this principle in a $4 \times 4 \times 4$ cube, where each point x has coordinates (x_1, x_2, x_3) with $0 \leq x_1, x_2, x_3 < 4$. The vertices of this cube form a median algebra because $\langle xyz \rangle = (\langle x_1 y_1 z_1 \rangle, \langle x_2 y_2 z_2 \rangle, \langle x_3 y_3 z_3 \rangle)$; furthermore, the edges of the graph in Fig. 7 are those defined in (61), running between vertices whose coordinates agree except that one coordinate changes by ± 1. Three typical intervals $[x \mathinner{.\,.} y]$, $[x \mathinner{.\,.} z]$, and $[y \mathinner{.\,.} z]$ are shown; the only point common to all three intervals is the vertex $\langle xyz \rangle = (2, 2, 1)$.



(a) The interval $[x \mathinner{.\,.} y]$. (b) The interval $[x \mathinner{.\,.} z]$. (c) The interval $[y \mathinner{.\,.} z]$.

Fig. 7. Intervals between the vertices $x = (0, 2, 1)$, $y = (3, 3, 3)$, and $z = (2, 0, 0)$ in a $4 \times 4 \times 4$ cube.

So far we've started with a median algebra and used it to define a graph with certain properties. But we can also start with a graph that has those properties and use it to define a median algebra. If u and v are vertices of *any* graph, let us define the interval $[u\,..\,v]$ to be the set of all points on shortest paths between u and v. A finite graph is said to be a *median graph* if exactly one vertex lies in the intersection $[x\,..\,y]\cap[x\,..\,z]\cap[y\,..\,z]$ of the three intervals that tie any three given vertices x, y, and z together; and we denote that vertex by $\langle xyz\rangle$. Exercise 75 proves that the resulting ternary operation satisfies the median axioms.

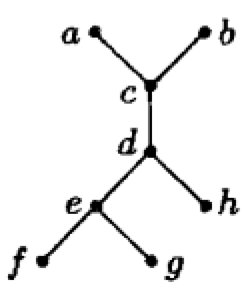
Many important graphs turn out to be median graphs according to this definition. For example, any free tree is easily seen to be a median graph; and a graph like the $n_1\times n_2\times\cdots\times n_m$ hyperrectangle provides another simple example. Cartesian products of arbitrary median graphs also satisfy the required condition.

***Median labels.** If u and v are any elements of a median algebra, the mapping $f(x)$ that takes $x\mapsto\langle xuv\rangle$ is a *homomorphism*; that is, it satisfies

$$f(\langle xyz\rangle) \;=\; \langle f(x)f(y)f(z)\rangle, \tag{62}$$

because of the long distributive law (54). This function $\langle xuv\rangle$ "projects" any given point x into the interval $[u\,..\,v]$, by (57). And it is particularly interesting in the case when $u — v$ is an edge of the corresponding graph, because $f(x)$ is then two-valued, essentially a Boolean mapping.

For example, consider the typical free tree shown below, with eight vertices and seven edges. We can project each vertex x onto each of the edge intervals $[u\,..\,v]$ by deciding whether x is closer to u or to v:



	ac	bc	cd	de	ef	eg	dh	
$a\mapsto$	a	c	c	d	e	e	d	0000000
$b\mapsto$	c	b	c	d	e	e	d	1100000
$c\mapsto$	c	c	c	d	e	e	d	1000000
$d\mapsto$	c	c	d	d	e	e	d	1010000
$e\mapsto$	c	c	d	e	e	e	d	1011000
$f\mapsto$	c	c	d	e	f	e	d	1011100
$g\mapsto$	c	c	d	e	e	g	d	1011010
$h\mapsto$	c	c	d	d	e	e	h	1010001

(63)

On the right we've reduced the projections to 0s and 1s, arbitrarily deciding that $a\mapsto 0000000$. The resulting bit strings are called *labels* of the vertices, and we write, for example, $l(b)=1100000$. Since each projection is a homomorphism, we can calculate the median of any three points by simply taking Boolean medians in each component of their labels. For example, to compute $\langle bgh\rangle$ we find the bitwise median of $l(b)=1100000$, $l(g)=1011010$, and $l(h)=1010001$, namely $1010000=l(d)$.

When we project onto all the edges of a median graph, we might find that two columns of the binary labels are identical. This situation cannot occur with a free tree, but let's consider what would happen if the edge $g — h$ were added to the tree in (63): The resulting graph would still be a median graph, but the

columns for eg and dh would become identical (except with $e \leftrightarrow d$ and $g \leftrightarrow h$). Furthermore, the new column for gh would turn out to be equivalent to the column for de. Redundant components should be omitted from the labels in such cases; therefore the vertices of the augmented graph would have six-bit labels, like $l(g) = 101101$ and $l(h) = 101001$, instead of seven-bit labels.

The elements of any median algebra can always be represented by labels in this way. Therefore *any identity that holds in the Boolean case will be true in all median algebras.* This "zero-one principle" makes it possible to test whether any two given expressions built from the ternary operation $\langle xyz\rangle$ can be shown to be equal as a consequence of axioms (50), (51), and (52)—although we do have to check $2^{n-1}-1$ cases when we test n-variable expressions by this method.

For example, the associative law $\langle xw\langle ywz\rangle\rangle = \langle\langle xwy\rangle wz\rangle$ suggests that there should be a symmetrical interpretation of both sides that does not involve nested brackets. And indeed, there is such a formula:

$$\langle xw\langle ywz\rangle\rangle = \langle\langle xwy\rangle wz\rangle = \langle xwywz\rangle, \tag{64}$$

where $\langle xwywz\rangle$ denotes the median of the five-element multiset $\{x,w,y,w,z\} = \{w,w,x,y,z\}$. We can prove this formula by using the zero-one principle, noting also that median is the same thing as majority in the Boolean case. In a similar way we can prove (49), and we can show that the function used by Post in (47) can be simplified to

$$\langle x_1y\langle x_2y\ldots y\langle x_{s-1}yx_s\rangle\ldots\rangle\rangle = \langle x_1yx_2y\ldots yx_{s-1}yx_s\rangle; \tag{65}$$

it's a median of $2s-1$ quantities, where nearly half of them are equal to y.

A set C of vertices in a graph is called *convex* if $[u\,..\,v] \subseteq C$ whenever $u \in C$ and $v \in C$. In other words, whenever the endpoints of a shortest path belong to C, all vertices of that path must also be present in C. (A convex set is therefore identical to what we called an "ideal," a few pages ago; now our language has become geometric instead of algebraic.) The *convex hull* of $\{v_1,\ldots,v_m\}$ is defined to be the smallest convex set that contains each of the vertices $v_1,\ldots,v_m$. Our theoretical results above have shown that every interval $[u\,..\,v]$ is convex; hence $[u\,..\,v]$ is the convex hull of the two-point set $\{u,v\}$. But in fact much more is true:

Theorem C. *The convex hull of $\{v_1,v_2,\ldots,v_m\}$ in a median graph is the set of all points*

$$C = \{\langle v_1xv_2x\ldots xv_m\rangle \mid x \in M\}. \tag{66}$$

Furthermore, x is in C if and only if $x = \langle v_1xv_2x\ldots xv_m\rangle$.

Proof. Clearly $v_j \in C$ for $1 \le j \le m$. Every point of C must belong to the convex hull, because the point $x' = \langle v_2x\ldots xv_m\rangle$ is in the hull (by induction on m), and because $\langle v_1x\ldots xv_m\rangle \in [v_1\,..\,x']$. The zero-one principle proves that

$$\langle x\langle v_1yv_2y\ldots yv_m\rangle\langle v_1zv_2z\ldots zv_m\rangle\rangle = \langle v_1\langle xyz\rangle v_2\langle xyz\rangle\ldots\langle xyz\rangle v_m\rangle; \tag{67}$$

hence C is convex. Setting $x = y$ in this formula proves that $\langle v_1xv_2x\ldots xv_m\rangle$ is the closest point of C to x, and that $\langle v_1xv_2x\ldots xv_m\rangle \in [x\,..\,z]$ for all $z \in C$. ∎

Corollary C. *Let the label of v_j be $v_{j1} \ldots v_{jt}$ for $1 \le j \le m$. Then the convex hull of $\{v_1, \ldots, v_m\}$ is the set of all $x \in M$ whose label $x_1 \ldots x_t$ satisfies $x_j = c_j$ whenever $v_{1j} = v_{2j} = \cdots = v_{mj} = c_j$.* ∎

For example, the convex hull of $\{c, g, h\}$ in (63) consists of all elements whose label matches the pattern 10**0**, namely $\{c, d, e, g, h\}$.

When a median graph contains a 4-cycle $u — x — v — y — u$, the edges $u — x$ and $v — y$ are equivalent, in the sense that projection onto $[u \,.\,.\, x]$ and projection onto $[v \,.\,.\, y]$ both yield the same label coordinates. The reason is that, for any z with $\langle zux \rangle = u$, we have

$$\begin{aligned} y = \langle uvy \rangle &= \langle \langle zux \rangle vy \rangle \\ &= \langle \langle zvy \rangle \langle uvy \rangle \langle xvy \rangle \rangle \\ &= \langle \langle zvy \rangle yv \rangle, \end{aligned}$$

hence $\langle zvy \rangle = y$; similarly $\langle zux \rangle = x$ implies $\langle zvy \rangle = v$. The edges $x — v$ and $y — u$ are equivalent for the same reasons. Exercise 77 shows, among other things, that two edges yield equivalent projections if and only if they can be proved equivalent by a chain of equivalences obtained from 4-cycles in this way. Therefore the number of bits in each vertex label is the number of equivalence classes of edges induced by the 4-cycles; and it follows that the reduced labels for vertices are uniquely determined, once we specify a vertex whose label is $00\ldots0$.

A nice way to find the vertex labels of any median graph was discovered by P. K. Jha and G. Slutzki [*Ars Combin.* **34** (1992), 75–92] and improved by J. Hagauer, W. Imrich, and S. Klavžar [*Theor. Comp. Sci.* **215** (1999), 123–136]:

Algorithm H (*Median labels*). Given a median graph G and a source vertex a, this algorithm determines the equivalence classes defined by the 4-cycles of G, and computes the labels $l(v) = v_1 \ldots v_t$ of each vertex, where t is the number of classes and $l(a) = 0\ldots0$.

H1. [Initialize.] Preprocess G by visiting all vertices in order of their distance from a. For each edge $u — v$, we say that u is an *early neighbor* of v if a is closer to u than to v, otherwise u is a *late neighbor*; in other words, the early neighbors of v will already have been visited when v is encountered, but the late neighbors will still be awaiting their turn. Rearrange all adjacency lists so that early neighbors are listed first. Place each edge initially in its own equivalence class; a "union-find algorithm" like Algorithm 2.3.3E will be used to merge classes when the algorithm learns that they're equivalent.

H2. [Call the subroutine.] Set $j \leftarrow 0$ and invoke Subroutine I with parameter a. (Subroutine I appears below. The global variable j will be used to create a master list of edges $r_j — s_j$ for $1 \le j < n$, where n is the total number of vertices; there will be one entry with $s_j = v$, for each vertex $v \ne a$.)

H3. [Assign the labels.] Number the equivalence classes from 1 to t. Then set $l(a)$ to the t-bit string $0\ldots0$. For $j = 1, 2, \ldots, n-1$ (in this order), set $l(s_j)$ to $l(r_j)$ with bit k changed from 0 to 1, where k is the equivalence class of edge $r_j — s_j$. ∎

Subroutine I (*Process descendants of r*). This recursive subroutine, with parameter r and global variable j, does the main work of Algorithm H on the graph of all vertices currently reachable from vertex r. In the course of processing, all such vertices will be recorded on the master list, except r itself, and all edges between them will be removed from the current graph. Each vertex has four fields called its LINK, MARK, RANK, and MATE, initially null.

I1. [Loop over s.] Choose a vertex s with r — s. If there is no such vertex, return from the subroutine.

I2. [Record the edge.] Set $j \leftarrow j+1$, $r_j \leftarrow r$, and $s_j \leftarrow s$.

I3. [Begin breadth-first search.] (Now we want to find and delete all edges of the current graph that are equivalent to r — s.) Set $\mathtt{MARK}(s) \leftarrow s$, $\mathtt{RANK}(s) \leftarrow 1$, $\mathtt{LINK}(s) \leftarrow \Lambda$, and $v \leftarrow q \leftarrow s$.

I4. [Find the mate of v.] Find the early neighbor u of v for which $\mathtt{MARK}(u) \neq s$. (There will be exactly one such vertex u. Recall that early neighbors have been placed first, in step H1.) Set $\mathtt{MATE}(v) \leftarrow u$.

I5. [Delete u — v.] Make the edges u — v and r — s equivalent by merging their equivalence classes. Remove u and v from each other's adjacency lists.

I6. [Classify the neighbors of v.] For each early neighbor u of v, do step I7; for each late neighbor u of v, do step I8. Then go to step I9.

I7. [Note a possible equivalence.] If $\mathtt{MARK}(u) = s$ and $\mathtt{RANK}(u) = 1$, make the edge u — v equivalent to the edge $\mathtt{MATE}(u)$ — $\mathtt{MATE}(v)$. Return to I6.

I8. [Rank u.] If $\mathtt{MARK}(u) = s$ and $\mathtt{RANK}(u) = 1$, return to I6. Otherwise set $\mathtt{MARK}(u) \leftarrow s$ and $\mathtt{RANK}(u) \leftarrow 2$. Set w to the first neighbor of u (it will be early). If $w = v$, reset w to u's second early neighbor; but return to I6 if u has only one early neighbor. If $\mathtt{MARK}(w) \neq s$ or $\mathtt{RANK}(w) \neq 2$, set $\mathtt{RANK}(u) \leftarrow 1$, $\mathtt{LINK}(u) \leftarrow \Lambda$, $\mathtt{LINK}(q) \leftarrow u$, and $q \leftarrow u$. Return to I6.

I9. [Continue breadth-first search.] Set $v \leftarrow \mathtt{LINK}(v)$. Return to I4 if $v \neq \Lambda$.

I10. [Process subgraph s.] Call Subroutine I recursively with parameter s. Then return to I1. ▮

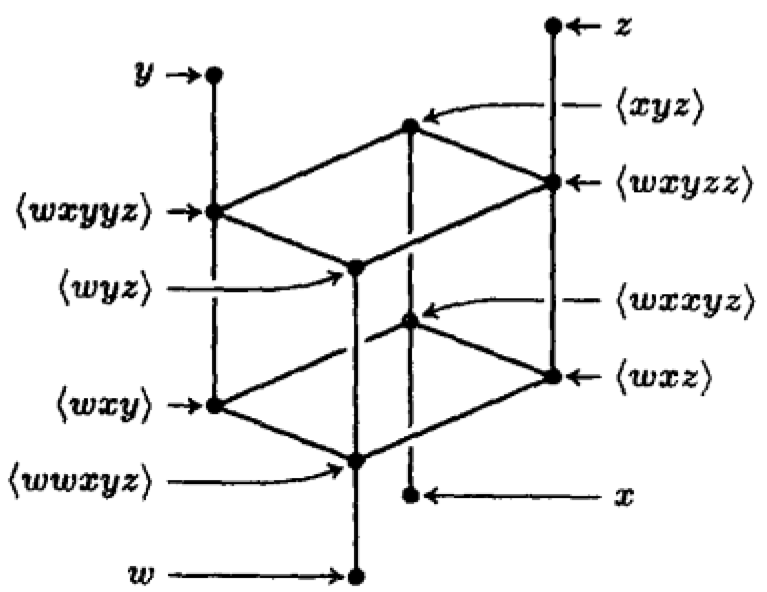
This algorithm and subroutine have been described in terms of relatively high-level data structures; further details are left to the reader. For example, adjacency lists should be doubly linked, so that edges can readily be deleted in step I5. Any convenient method for merging equivalence classes can be used in that step.

Exercise 77 explains the theory that makes this algorithm work, and exercise 78 proves that each vertex is encountered at most $\lg n$ times in step I4. Furthermore, exercise 79 shows that a median graph has at most $O(n \log n)$ edges. Therefore the total running time of Algorithm H is $O(n(\log n)^2)$, except perhaps for the bit-setting in step H3.

The reader may wish to play through Algorithm H by hand on the median graph in Table 2, whose vertices represent the twelve monotone self-dual Boolean functions of four variables $\{w, x, y, z\}$. All such functions that actually involve all four variables can be expressed as a median of five things, like (64). With

Table 2

LABELS FOR THE FREE MEDIAN ALGEBRA ON FOUR GENERATORS



j	r_j	s_j	$l(s_j)$
		w	0000000
1	w	$\langle wwxyz\rangle$	0000001
2	$\langle wwxyz\rangle$	$\langle wyz\rangle$	0010001
3	$\langle wyz\rangle$	$\langle wxyzz\rangle$	0010101
4	$\langle wxyzz\rangle$	$\langle xyz\rangle$	0010111
5	$\langle wxyzz\rangle$	z	1010101
6	$\langle wyz\rangle$	$\langle wxyyz\rangle$	0010011
7	$\langle wxyyz\rangle$	y	0110011
8	$\langle wwxyz\rangle$	$\langle wxz\rangle$	0000101
9	$\langle wxz\rangle$	$\langle wxxyz\rangle$	0000111
10	$\langle wxxyz\rangle$	x	0001111
11	$\langle wwxyz\rangle$	$\langle wxy\rangle$	0000011

starting vertex $a = w$, the algorithm computes the master list of edges $r_j — s_j$ and the binary labels shown in the table. (The actual order of processing depends on the order in which vertices appear in adjacency lists. But the final labels will be the same under any ordering, except for permutations of the columns.)

Notice that the number of 1-bits in each label $l(v)$ is the distance of v from the starting vertex a. In fact, the uniqueness of labels tells us that *the distance between any two vertices is the number of bit positions in which their labels differ*, because we could have started at any particular vertex.

The special median graph in Table 2 could actually have been handled in a completely different way, without using Algorithm H at all, because the labels in this case are essentially the same as the *truth tables* of the corresponding functions. Here's why: We can say that the simple functions w, x, y, z have the respective truth tables $t(w) = 0000000011111111$, $t(x) = 0000111100001111$, $t(y) = 0011001100110011$, $t(z) = 0101010101010101$. Then the truth table of $\langle wwxyz\rangle$ is the bitwise majority function $\langle t(w)t(w)t(x)t(y)t(z)\rangle$, namely the string 0000000101111111; and a similar computation gives the truth tables of all the other vertices.

The last half of any self-dual function's truth table is the same as the first half, but complemented and reversed, so we can eliminate it. Furthermore the leftmost bit in each of our truth tables is always zero. We are left with the seven-bit labels shown in Table 2; and the uniqueness property guarantees that Algorithm H will produce the same result, except for possible permutation of columns, when it is presented with this particular graph.

This reasoning tells us that the edges of the graph in Table 2 correspond to pairs of functions whose truth tables are almost the same. We move between neighboring vertices by switching only two complementary bits of their truth tables. In fact, the degree of each vertex turns out to be exactly the number of prime implicants in the disjunctive prime form of the monotone self-dual function represented by that vertex (see exercises 70 and 84).

***Median sets.** A *median set* is a collection X of binary vectors with the property that $\langle xyz\rangle \in X$ whenever $x \in X$, $y \in X$, and $z \in X$, where the medians are computed componentwise as we've done with median labels. Thomas Schaefer noticed in 1978 that median sets provide us with an attractive counterpoint to the characterization of Horn functions in Theorem H:

Theorem S. *The Boolean function $f(x_1,\ldots,x_n)$ is expressible as a conjunction of Krom clauses if and only if*

$$f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n) = f(z_1,\ldots,z_n) = 1 \quad \text{implies} \quad f(\langle x_1y_1z_1\rangle,\ldots,\langle x_ny_nz_n\rangle) = 1 \qquad (68)$$

for all Boolean values x_j, y_j, and z_j.

Proof. [*STOC* **10** (1978), 216–226, Lemma 3.1B.] If we have $x_1 \vee x_2 = y_1 \vee y_2 = z_1 \vee z_2 = 1$, say, with $x_1 \le y_1 \le z_1$, then $\langle x_1y_1z_1\rangle \vee \langle x_2y_2z_2\rangle = y_1 \vee \langle x_2y_2z_2\rangle = 1$, since $y_1 = 0$ implies that $x_2 = y_2 = 1$. Thus (68) is necessary.

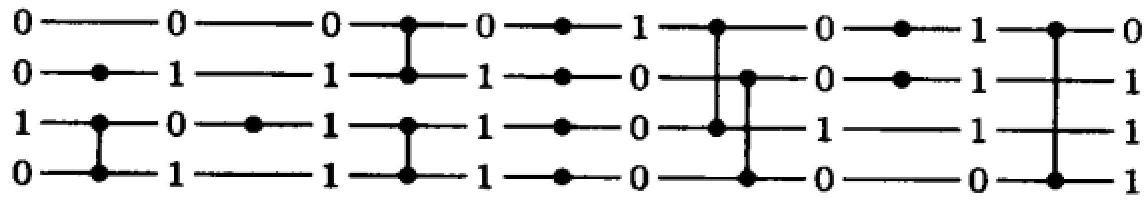
Conversely, if (68) holds, let $u_1 \vee \cdots \vee u_k$ be a prime clause of f, where each u_j is a literal. Then, for $1 \le j \le k$, the clause $u_1 \vee \cdots \vee u_{j-1} \vee u_{j+1} \vee \cdots \vee u_k$ is not a clause of f; so there's a vector $x^{(j)}$ with $f(x^{(j)}) = 1$ but with $u_i^{(j)} = 0$ for all $i \ne j$. If $k \ge 3$, the median $\langle x^{(1)}x^{(2)}x^{(3)}\rangle$ has $u_i = 0$ for $1 \le i \le k$; but that's impossible, because $u_1 \vee \cdots \vee u_k$ was supposedly a clause. Hence $k \le 2$. ∎

Thus median sets are the same as "2SAT instances," the sets of points that satisfy some formula f in 2CNF.

A median set is said to be *reduced* if its vectors $x = x_1 \ldots x_t$ contain no redundant components. In other words, for each coordinate position k, a reduced median set has at least two vectors $x^{(k)}$ and $y^{(k)}$ with the property that $x_k^{(k)} = 0$ and $y_k^{(k)} = 1$ but $x_i^{(k)} = y_i^{(k)}$ for all $i \ne k$. We've seen that the labels of a median graph satisfy this condition; in fact, if coordinate k corresponds to the edge u — v in the graph, we can let $x^{(k)}$ and $y^{(k)}$ be the labels of u and v. Conversely, any reduced median set X defines a median graph, with one vertex for each element of X and with adjacency defined by all-but-one equality of coordinates. The median labels of these vertices must be identical to the original vectors in X, because we know that median labels are essentially unique.

Median labels and reduced median sets can also be characterized in yet another instructive way, which harks back to the networks of *comparator modules* that we studied in Section 5.3.4. We noted in that section that such networks are useful for "oblivious sorting" of numbers, and we noted in Theorem 5.3.4Z that a network of comparators will sort all $n!$ possible input permutations if and only if it correctly sorts all 2^n combinations of 0s and 1s. When a comparator module is attached to two horizontal lines, with inputs x and y entering from the left, it outputs the same two values on the right, but with $\min(x,y) = x \wedge y$ on the upper line and $\max(x,y) = x \vee y$ on the lower line. Let's now extend the concept slightly by also allowing *inverter modules*, which change 0 to 1 and vice versa. Here, for example, is a comparator-inverter network (or CI-net, for

short), which transforms the binary value 0010 into 0111:



(69)

(A single dot denotes an inverter.) Indeed, this network transforms

$$\begin{array}{llll} 0000 \mapsto 0110; & 0100 \mapsto 0111; & 1000 \mapsto 0111; & 1100 \mapsto 0110; \\ 0001 \mapsto 0111; & 0101 \mapsto 1111; & 1001 \mapsto 0101; & 1101 \mapsto 0111; \\ 0010 \mapsto 0111; & 0110 \mapsto 1111; & 1010 \mapsto 0101; & 1110 \mapsto 0111; \\ 0011 \mapsto 0110; & 0111 \mapsto 0111; & 1011 \mapsto 0111; & 1111 \mapsto 0110. \end{array} \qquad (70)$$

Suppose a CI-net transforms the bit string $x = x_1 \ldots x_t$ into the bit string $x'_1 \ldots x'_t = f(x)$. This function f, which maps the t-cube into itself, is in fact a *graph homomorphism*. In other words, we have $f(x) \mathbin{—} f(y)$ whenever $x \mathbin{—} y$ in the t-cube: Changing one bit of x always causes exactly one bit of $f(x)$ to change, because every module in the network has this behavior. Moreover, CI-nets have a remarkable connection with median labels:

Theorem F. *Every set X of t-bit median labels can be represented by a comparator-inverter network that computes a Boolean function $f(x)$ with the property that $f(x) \in X$ for all bit vectors $x_1 \ldots x_t$, and $f(x) = x$ for all $x \in X$.*

Proof. [Tomás Feder, *Memoirs Amer. Math. Soc.* **555** (1995), 1–223, Lemma 3.37; see also the Ph. D. thesis of D. H. Wiedemann (University of Waterloo, 1986).] Consider columns i and j of the median labels, where $1 \le i < j \le t$. Any such pair of columns contains at least three of the four possibilities $\{00, 01, 10, 11\}$, if we look through the entire set of labels, because median labels have no redundant columns. Let us write $\bar{\jmath} \to i$, $j \to i$, $i \to j$, or $i \to \bar{\jmath}$ if the value 00, 01, 10, or 11 (respectively) is missing from those two columns; we can also note the equivalent relations $\bar{\imath} \to j$, $\bar{\imath} \to \bar{\jmath}$, $\bar{\jmath} \to \bar{\imath}$, or $j \to \bar{\imath}$, respectively, which involve $\bar{\imath}$ instead of i. For example, the labels in Table 2 give us the relations

$$\begin{array}{ll} 1 \to \bar{2}, 3, \bar{4}, 5, \bar{6}, 7 & 2, \bar{3}, 4, \bar{5}, 6, \bar{7} \to \bar{1}; \\ 2 \to 3, \bar{4}, \bar{5}, 6, 7 & \bar{3}, 4, 5, \bar{6}, \bar{7} \to \bar{2}; \\ 3 \to \bar{4}, 7 & 4, \bar{7} \to \bar{3}; \\ 4 \to 5, 6, 7 & \bar{5}, \bar{6}, \bar{7} \to \bar{4}; \\ 5 \to 7 & \bar{7} \to \bar{5}; \\ 6 \to 7 & \bar{7} \to \bar{6}. \end{array} \qquad (71)$$

(There is no relation between 3 and 5 because all four possibilities occur in those columns. But we have $3 \to \bar{4}$ because 11 doesn't appear in columns 3 and 4. The vertices whose label has a 1 in column 3 are those closer to $\langle wyz \rangle$ than to $\langle wwxyz \rangle$ in Table 2; they form a convex set in which column 4 of the labels is always 0, because they are also closer to $\langle wxxyz \rangle$ than to x.)

These relations between the literals $\{1, \bar{1}, 2, \bar{2}, \ldots, t, \bar{t}\}$ contain no cycles, so they can always be topologically sorted into an anti-symmetrical sequence

$u_1 u_2 \ldots u_{2t}$ in which u_j is the complement of u_{2t+1-j}. For example,

$$1\ \bar{7}\ 4\ 2\ \bar{3}\ \bar{5}\ \bar{6}\ 6\ 5\ 3\ \bar{2}\ \bar{4}\ 7\ \bar{1} \qquad (72)$$

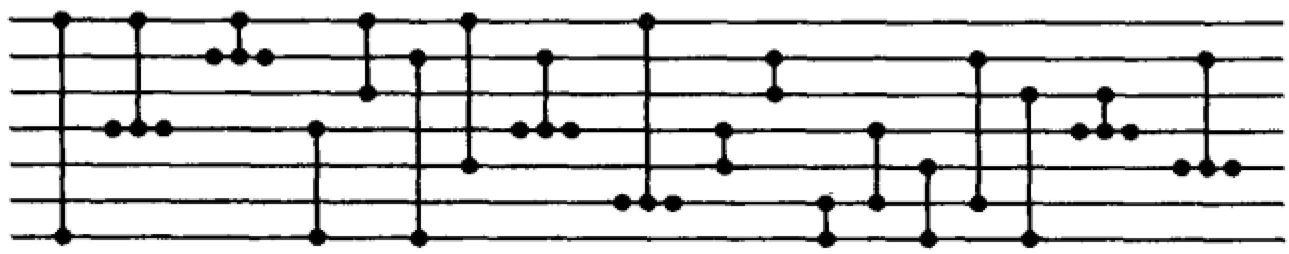
is one such way to sort the relations in (71) topologically.

Now we proceed to construct the network, by starting with t empty lines and successively examining elements u_k and u_{k+d} in the topological sequence, for $d = 2t-2,\ 2t-3,\ \ldots,\ 1$ (in this order), and for $k = 1, 2, \ldots, t - \lceil d/2 \rceil$. If $u_k \le u_{k+d}$ is a relation between columns i and j, where $i < j$, we append new modules to lines i and j of the network as follows:



(73)

For example, from (71) and (72) we first enforce $1 \to 7$, then $1 \to \bar{4}$, then $1 \to \bar{2}$, then $\bar{7} \to \bar{4}$ (that is, $4 \to 7$), etc., obtaining the following network:


(74)

(Go figure. No modules are contributed when, say, u_k is $\bar{7}$ and u_{k+d} is 3, because the relation $\bar{3} \to 7$ does not appear in (71).)

Exercise 89 proves that each new cluster of modules (73) preserves all of the previous inequalities and enforces a new one. Therefore, if x is any input vector, $f(x)$ satisfies all of the inequalities; so $f(x) \in X$ by Theorem S. Conversely, if $x \in X$, every cluster of modules in the network leaves x unchanged. ▮

Corollary F. *Suppose the median labels in Theorem F are closed under the operations of bitwise AND and OR, so that $x \,\&\, y \in X$ and $x \mid y \in X$ whenever $x \in X$ and $y \in X$. Then there is a permutation of coordinates under which the labels are representable by a network of comparator modules only.*

Proof. The bitwise AND of all labels is $0\ldots0$, and the bitwise OR is $1\ldots1$. So the only possible relations between columns are $i \to j$ and $j \to i$. By topologically sorting and renaming the columns, we can ensure that only $i \to j$ occurs when $i < j$; and in this case the construction in the proof never uses an inverter. ▮

In general, if G is any graph, a homomorphism f that maps the vertices of G onto a subset X of those vertices is called a *retraction* if it satisfies $f(x) = x$ for all $x \in X$; and we call X a *retract* of G when such an f exists. The importance of this concept in the theory of graphs was first pointed out by Pavol Hell [see *Lecture Notes in Math.* **406** (1974), 291–301]. One consequence, for example, is that the distance between vertices in X — the number of edges on a shortest path — remains the same even if we restrict consideration to paths that lie entirely in X. (See exercise 93.)

Theorem F demonstrates that every t-dimensional set of median labels is a retract of the t-dimensional hypercube. Conversely, exercise 94 shows that hypercube retracts are always median graphs.

Threshold functions. A particularly appealing and important class of Boolean functions $f(x_1, x_2, \ldots, x_n)$ arises when f can be defined by the formula

$$f(x_1, x_2, \ldots, x_n) = [w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq t], \tag{75}$$

where the constants w_1, w_2, …, w_n are integer "weights" and t is an integer "threshold" value. For example, threshold functions are important even when all the weights are unity: We have

$$x_1 \wedge x_2 \wedge \cdots \wedge x_n = [x_1 + x_2 + \cdots + x_n \geq n]; \tag{76}$$

$$x_1 \vee x_2 \vee \cdots \vee x_n = [x_1 + x_2 + \cdots + x_n \geq 1]; \tag{77}$$

and

$$\langle x_1x_2 \ldots x_{2t-1}\rangle = [x_1 + x_2 + \cdots + x_{2t-1} \geq t], \tag{78}$$

where $\langle x_1x_2 \ldots x_{2t-1}\rangle$ stands for the median (or majority) value of a multiset that consists of any odd number of Boolean values $\{x_1, x_2, \ldots, x_{2t-1}\}$. In particular, the basic mappings $x \wedge y$, $x \vee y$, and $\langle xyz\rangle$ are all threshold functions, and so is

$$\bar{x} = [-x \geq 0]. \tag{79}$$

With more general weights we get many other functions of interest, such as

$$[2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + x_n \geq (t_1t_2 \ldots t_n)_2], \tag{80}$$

which is true if and only if the binary string $x_1x_2 \ldots x_n$ is lexicographically greater than or equal to a given binary string $t_1t_2 \ldots t_n$. Given a set of n objects having sizes w_1, w_2, …, w_n, a subset of those objects will fit into a knapsack of size $t - 1$ if and only if $f(x_1, x_2, \ldots, x_n) = 0$, where $x_j = 1$ represents the presence of object j in the subset. Simple models of neurons, originally proposed by W. McCullough and W. Pitts in *Bull. Math. Biophysics* **5** (1943), 115–133, have led to thousands of research papers about "neural networks" built from threshold functions.

We can get rid of any negative weight w_j by setting $x_j \leftarrow \bar{x}_j$, $w_j \leftarrow -w_j$, and $t \leftarrow t + |w_j|$. Thus a general threshold function can be reduced to a positive threshold function in which all weights are nonnegative. Furthermore, any positive threshold function (75) can be expressed as a special case of the median/majority-of-odd function, because we have

$$\langle 0^a1^bx_1^{w_1}x_2^{w_2} \ldots x_n^{w_n}\rangle = [b + w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq b + t], \tag{81}$$

where x^m stands for m copies of x, and where a and b are defined by the rules

$$a = \max(0, 2t-1-w), \quad b = \max(0, w+1-2t), \quad w = w_1+w_2+\cdots+w_n. \tag{82}$$

For example, when all weights are 1, we have

$$\langle 0^{n-1}x_1 \ldots x_n\rangle = x_1 \wedge \cdots \wedge x_n \quad \text{and} \quad \langle 1^{n-1}x_1 \ldots x_n\rangle = x_1 \vee \cdots \vee x_n; \tag{83}$$

we've already seen these formulas in (45) when $n = 2$. In general, either a or b is zero, and the left-hand side of (81) specifies a median of $2T - 1$ elements, where

$$T = b + t = \max(t, w_1 + w_2 + \cdots + w_n + 1 - t). \tag{84}$$

There would be no point in letting both a and b be greater than zero, because the majority function clearly satisfies the cancellation law

$$\langle 01x_1x_2\ldots x_{2t-1}\rangle \;=\; \langle x_1x_2\ldots x_{2t-1}\rangle. \tag{85}$$

One important consequence of (81) is that every positive threshold function comes from the pure majority function

$$g(x_0,x_1,x_2,\ldots,x_n) \;=\; \langle x_0^{a+b}x_1^{w_1}x_2^{w_2}\ldots x_n^{w_n}\rangle \tag{86}$$

by setting $x_0 = 0$ or 1. In other words, we know all threshold functions of n variables if and only if we know all of the distinct median-of-odd functions of $n+1$ or fewer variables (containing no constants). Every pure majority function is monotone and self-dual; thus we've seen the pure majority functions of four variables $\{w,x,y,z\}$ in column s_j of Table 2 on page 71, namely $\langle w\rangle$, $\langle wwxyz\rangle$, $\langle wyz\rangle$, $\langle wxyzz\rangle$, $\langle xyz\rangle$, $\langle z\rangle$, $\langle wxyyz\rangle$, $\langle y\rangle$, $\langle wxz\rangle$, $\langle wxxyz\rangle$, $\langle x\rangle$, $\langle wxy\rangle$. By setting $w=0$ or 1, we obtain all the positive threshold functions $f(x,y,z)$ of three variables:

$$\begin{aligned}&\langle 0\rangle,\langle 1\rangle,\langle 00xyz\rangle,\langle 11xyz\rangle,\langle 0yz\rangle,\langle 1yz\rangle,\langle 0xyzz\rangle,\langle 1xyzz\rangle,\langle xyz\rangle,\langle z\rangle,\\ &\quad\langle 0xyyz\rangle,\langle 1xyyz\rangle,\langle y\rangle,\langle 0xz\rangle,\langle 1xz\rangle,\langle 0xxyz\rangle,\langle 1xxyz\rangle,\langle x\rangle,\langle 0xy\rangle,\langle 1xy\rangle.\end{aligned} \tag{87}$$

All 150 positive threshold functions of four variables can be obtained in a similar fashion from the self-dual majority functions in the answer to exercise 84.

There are infinitely many sequences of weights $(w_1,w_2,\ldots,w_n)$, but only finitely many threshold functions for any given value of n. So it is clear that many different weight sequences are equivalent. For example, consider the pure majority function

$$\langle x_1^2x_2^3x_3^5x_4^7x_5^{11}x_6^{13}\rangle,$$

in which prime numbers have been used as weights. A brute-force examination of 2^6 cases shows that

$$\langle x_1^2x_2^3x_3^5x_4^7x_5^{11}x_6^{13}\rangle = \langle x_1x_2^2x_3^2x_4^3x_5^4x_6^5\rangle; \tag{88}$$

thus we can express the same function with substantially smaller weights. Similarly, the threshold function

$$[(x_1x_2\ldots x_{20})_2 \ge (01100100100001111110)_2] = \langle 1^{225028}x_1^{524288}x_2^{262144}\ldots x_{20}\rangle,$$

a special case of (80), turns out to be simply

$$\langle 1^{323}x_1^{764}x_2^{323}x_3^{323}x_4^{118}x_5^{118}x_6^{87}x_7^{31}x_8^{31}x_9^{25}x_{10}^6x_{11}^6x_{12}^6x_{13}^6x_{14}x_{15}x_{16}x_{17}x_{18}x_{19}\rangle. \tag{89}$$

Exercise 103 explains how to find a minimum set of weights without resorting to a huge brute-force search, using linear programming.

A nice indexing scheme by which a unique identifier can be assigned to any threshold function was discovered by C. K. Chow [*FOCS* **2** (1961), 34–38]. Given any Boolean function $f(x_1,\ldots,x_n)$, let $N(f)$ be the number of vectors $x=(x_1,\ldots,x_n)$ for which $f(x)=1$, and let $\Sigma(f)$ be the sum of all those vectors. For example, if $f(x_1,x_2)=x_1\vee x_2$, we have $N(f)=3$ and $\Sigma(f)=(0,1)+(1,0)+(1,1)=(2,2)$.

Theorem T. *Let $f(x_1,\ldots,x_n)$ and $g(x_1,\ldots,x_n)$ be Boolean functions with $N(f)=N(g)$ and $\Sigma(f)=\Sigma(g)$, where f is a threshold function. Then $f=g$.*

Proof. Suppose there are exactly k vectors $x^{(1)}, \ldots, x^{(k)}$ such that $f(x^{(j)})=1$ and $g(x^{(j)})=0$. Since $N(f)=N(g)$, there must be exactly k vectors $y^{(1)}, \ldots, y^{(k)}$ such that $f(y^{(j)})=0$ and $g(y^{(j)})=1$. And since $\Sigma(f)=\Sigma(g)$, we must also have $x^{(1)}+\cdots+x^{(k)}=y^{(1)}+\cdots+y^{(k)}$.

Now suppose f is the threshold function (75); then we have $w\cdot x^{(j)}\ge t$ and $w\cdot y^{(j)}<t$ for $1\le j\le k$. But if $f\ne g$ we have $k>0$, and $w\cdot(x^{(1)}+\cdots+x^{(k)})\ge kt>w\cdot(y^{(1)}+\cdots+y^{(k)})$, a contradiction. ∎

Threshold functions have many curious properties, some of which are explored in the exercises below. Their classical theory is well summarized in Saburo Muroga's book *Threshold Logic and its Applications* (Wiley, 1971).

Symmetric Boolean functions. A function $f(x_1,\ldots,x_n)$ is called *symmetric* if $f(x_1,\ldots,x_n)$ is equal to $f(x_{p(1)},\ldots,x_{p(n)})$ for all permutations $p(1)\ldots p(n)$ of $\{1,\ldots,n\}$. When all the x_j are 0 or 1, this condition means that f depends only on the number of 1s that are present in the arguments, namely the "sideways sum" $\nu x=\nu(x_1,\ldots,x_n)=x_1+\cdots+x_n$. The notation $S_{k_1,k_2,\ldots,k_r}(x_1,\ldots,x_n)$ is commonly used to stand for the Boolean function that is true if and only if νx is either k_1 or k_2 or $\cdots$ or k_r. For example, $S_{1,3,5}(v,w,x,y,z)=v\oplus w\oplus x\oplus y\oplus z$; $S_{3,4,5}(v,w,x,y,z)=\langle vwxyz\rangle$; $S_{4,5}(v,w,x,y,z)=\langle 00vwxyz\rangle$.

Many applications of symmetry involve the basic functions $S_k(x_1,\ldots,x_n)$ that are true only when $\nu x=k$. For example, $S_3(x_1,x_2,x_3,x_4,x_5,x_6)$ is true if and only if exactly half of the arguments $\{x_1,\ldots,x_6\}$ are true and the other half are false. In such cases we obviously have

$$S_k(x_1,\ldots,x_n) \;=\; S_{\ge k}(x_1,\ldots,x_n)\wedge\overline{S_{\ge k+1}(x_1,\ldots,x_n)}, \qquad (90)$$

where $S_{\ge k}(x_1,\ldots,x_n)$ is an abbreviation for $S_{k,k+1,\ldots,n}(x_1,\ldots,x_n)$. The functions $S_{\ge k}(x_1,\ldots,x_n)$ are, of course, the threshold functions $[x_1+\cdots+x_n\ge k]$ that we have already studied.

More complicated cases can be treated as threshold functions of threshold functions. For example, we have

$$\begin{aligned} S_{2,3,6,8,9}(x_1,\ldots,x_{12}) &= \big[\nu x\ge 2+4[\nu x\ge 4]+2[\nu x\ge 7]+5[\nu x\ge 10]\big] \\ &= \big\langle 00x_1\ldots x_{12}\langle 0^5\bar{x}_1\ldots\bar{x}_{12}\rangle^4\langle 1\bar{x}_1\ldots\bar{x}_{12}\rangle^2\langle 1^7\bar{x}_1\ldots\bar{x}_{12}\rangle^5\big\rangle, \end{aligned} \qquad (91)$$

because the number of 1s in the outermost majority-of-25 turns out to be respectively $(11,12,13,14,11,12,13,12,13,14,10,11,12)$ when $x_1+\cdots+x_{12}=(0,1,\ldots,12)$. A similar two-level scheme works in general [R. C. Minnick, *IRE Trans.* **EC-10** (1961), 6–16]; and with three or more levels of logic we can reduce the number of thresholding operations even further. (See exercise 113.)

A variety of ingenious tricks have been discovered for evaluating symmetric Boolean functions. For example, S. Muroga attributes the following remarkable sequence of formulas to F. Sasaki:

$$x_0\oplus x_1\oplus\cdots\oplus x_{2m}=\langle\bar{x}_0s_1s_2\ldots s_{2m}\rangle,$$
$$\text{where}\quad s_j=\langle x_0x_jx_{j+1}\ldots x_{j+m-1}\bar{x}_{j+m}\bar{x}_{j+m+1}\ldots\bar{x}_{j+2m-1}\rangle, \qquad (92)$$

if $m > 0$ and if we consider x_{2m+k} to be the same as x_k for $k \geq 1$. In particular, when $m = 1$ and $m = 2$ we have the identities

$$x_0 \oplus x_1 \oplus x_2 = \langle \bar{x}_0 \langle x_0 x_1 \bar{x}_2 \rangle \langle x_0 x_2 \bar{x}_1 \rangle \rangle; \tag{93}$$

$$x_0 \oplus \cdots \oplus x_4 = \langle \bar{x}_0 \langle x_0 x_1 x_2 \bar{x}_3 \bar{x}_4 \rangle \langle x_0 x_2 x_3 \bar{x}_4 \bar{x}_1 \rangle \langle x_0 x_3 x_4 \bar{x}_1 \bar{x}_2 \rangle \langle x_0 x_4 x_1 \bar{x}_2 \bar{x}_3 \rangle \rangle. \tag{94}$$

The right-hand sides are fully symmetric, but not obviously so! (See exercise 115.)

Canalizing functions. A Boolean function $f(x_1, \ldots, x_n)$ is said to be *canalizing* or "forcing" if we might be able to deduce its value by examining at most one of its variables. More precisely, f is canalizing if $n = 0$ or if there's a subscript j for which $f(x)$ either has a constant value when we set $x_j = 0$ or a constant value when we set $x_j = 1$. For example, $f(x, y, z) = (x \oplus z) \vee \bar{y}$ is canalizing because it always equals 1 when $y = 0$. (When $y = 1$ we don't know the value of f without examining also x and z; but half a loaf is better than none.) Such functions, introduced by Stuart Kauffman [*Lectures on Mathematics in the Life Sciences* **3** (1972), 63–116; *J. Theoretical Biology* **44** (1974), 167–190], have proved to be important in many applications, especially in chemistry and biology. Some of their properties are examined in exercises 125–129.

Quantitative considerations. We've been studying many different kinds of Boolean functions, so it's natural to ask: How many n-variable functions of each type actually exist? Tables 3, 4, and 5 provide the answers, at least for small values of n.

All functions are counted in Table 3. There are 2^{2^n} possibilities for each n, since there are 2^{2^n} possible truth tables. Some of these functions are self-dual, some are monotone; some are both monotone and self-dual, as in Theorem P. Some are Horn functions as in Theorem H; some are Krom functions as in Theorem S; and so on.

But in Table 4, two functions are considered identical if they differ only because the names of variables have changed. Thus only 12 different cases arise when $n = 2$, because (for example) $x \vee \bar{y}$ and $\bar{x} \vee y$ are essentially the same.

Table 5 goes a step further: It allows us to complement individual variables, and even to complement the entire function, without essentially changing it. From this perspective the 256 Boolean functions of (x, y, z) fall into only 14 different equivalence classes:

Representative	Class size	Representative	Class size
0	2	$x \wedge (y \oplus z)$	24
x	6	$x \oplus (y \wedge z)$	24
$x \wedge y$	24	$(x \wedge y) \vee (\bar{x} \wedge z)$	24
$x \oplus y$	6	$(x \vee y) \wedge (x \oplus z)$	48
$x \wedge y \wedge z$	16	$(x \oplus y) \vee (x \oplus z)$	8
$x \oplus y \oplus z$	2	$\langle xyz \rangle$	8
$x \wedge (y \vee z)$	48	$S_1(x, y, z)$	16

(95)

We shall study ways to count and to list inequivalent combinatorial objects in Section 7.2.3.

Table 3
BOOLEAN FUNCTIONS OF n VARIABLES

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
arbitrary	2	4	16	256	65,536	4,294,967,296	18,446,744,073,709,551,616
self-dual	0	2	4	16	256	65,536	4,294,967,296
monotone	2	3	6	20	168	7,581	7,828,354
both	0	1	2	4	12	81	2,646
Horn	2	4	14	122	4,960	2,771,104	151,947,502,948
Krom	2	4	16	166	4,170	224,716	24,445,368
threshold	2	4	14	104	1,882	94,572	15,028,134
symmetric	2	4	8	16	32	64	128
canalizing	2	4	14	120	3,514	1,292,276	103,071,426,294

Table 4
BOOLEAN FUNCTIONS DISTINCT UNDER PERMUTATION OF VARIABLES

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
arbitrary	2	4	12	80	3,984	37,333,248	25,626,412,338,274,304
self-dual	0	2	2	8	32	1,088	6,385,408
monotone	2	3	5	10	30	210	16,353
both	0	1	1	2	3	7	30
Horn	2	4	10	38	368	29,328	216,591,692
Krom	2	4	12	48	308	3,028	49,490
threshold	2	4	10	34	178	1,720	590,440
canalizing	2	4	10	38	294	15,774	149,325,022

Table 5
BOOLEAN FUNCTIONS DISTINCT UNDER COMPLEMENTATION/PERMUTATION

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
arbitrary	1	2	4	14	222	616,126	200,253,952,527,184
self-dual	0	1	1	3	7	83	109,950
threshold	1	2	3	6	15	63	567
both	0	1	1	2	3	7	21
canalizing	1	2	3	6	22	402	1,228,158

EXERCISES

1. [*15*] (Lewis Carroll.) Make sense of Tweedledee's comment, quoted near the beginning of this section. [*Hint:* See Table 1.]

2. [*17*] Logicians on the remote planet Pincus use the symbol 1 to represent "false" and 0 to represent "true." Thus, for example, they have a binary operation called "or" whose properties

$$1 \text{ or } 1 = 1, \qquad 1 \text{ or } 0 = 0, \qquad 0 \text{ or } 1 = 0, \qquad 0 \text{ or } 0 = 0$$

we associate with $\wedge$. What operations would we associate with the 16 logical operators that Pincusians respectively call "falsehood," "and," ..., "nand," "validity" (see Table 1)?

▶ **3.** [*13*] Suppose logical values were respectively -1 for falsehood and $+1$ for truth, instead of 0 and 1. What operations $\circ$ in Table 1 would then correspond to (a) $\max(x, y)$? (b) $\min(x, y)$? (c) $-x$? (d) $x \cdot y$?

4. [*24*] (H. M. Sheffer.) The purpose of this exercise is to show that all of the operations in Table 1 can be expressed in terms of NAND. (a) For each of the 16 operators $\circ$ in that table, find a formula equivalent to $x \circ y$ that uses only $\overline{\wedge}$ as an operator. Your formula should be as short as possible. For example, the answer for operation $\llcorner$ is simply "x", but the answer for $\overline{\llcorner}$ is "$x \mathbin{\overline{\wedge}} x$". Do not use the constants 0 or 1 in your formulas. (b) Similarly, find 16 short formulas when constants *are* allowed. For example, $x \mathbin{\overline{\llcorner}} y$ can now be expressed also as "$x \mathbin{\overline{\wedge}} 1$".

5. [*24*] Consider exercise 4 with $\overline{\subset}$ as the basic operation instead of $\overline{\wedge}$.

6. [*21*] (E. Schröder.) (a) Which of the 16 operations in Table 1 are associative—in other words, which of them satisfy $x \circ (y \circ z) = (x \circ y) \circ z$? (b) Which of them satisfy the identity $(x \circ y) \circ (y \circ z) = x \circ z$?

7. [*20*] Which operations in Table 1 have the property that $x \circ y = z$ if and only if $y \circ z = x$?

8. [*24*] Which of the 16^2 pairs of operations $(\circ, \square)$ satisfy the left-distributive law $x \circ (y \mathbin{\square} z) = (x \circ y) \mathbin{\square} (x \circ z)$?

9. [*16*] True or false? (a) $(x \oplus y) \vee z = (x \vee z) \oplus (y \vee z)$; (b) $(w \oplus x \oplus y) \vee z = (w \vee z) \oplus (x \vee z) \oplus (y \vee z)$; (c) $(x \oplus y) \vee (y \oplus z) = (x \oplus z) \vee (y \oplus z)$.

10. [*17*] What is the multilinear representation of the "random" function (22)?

11. [*M25*] Is there an intuitive way to understand exactly when the multilinear representation of $f(x_1, \ldots, x_n)$ contains, say, the term $x_2x_3x_6x_8$? (See (19).)

▶ **12.** [*M23*] The *integer multilinear representation* of a Boolean function extends representations like (19) to a polynomial $f(x_1, \ldots, x_n)$ with integer coefficients, in such a way that $f(x_1, \ldots, x_n)$ has the correct value (0 or 1) for all 2^n possible 0–1 vectors $(x_1, \ldots, x_n)$, *without* taking a remainder mod 2. For example, the integer multilinear representation corresponding to (19) is $1 - xy - xz - yz + 3xyz$.

a) What is the integer multilinear representation of the "random" function (22)?
b) How large can the coefficients of such a representation $f(x_1, \ldots, x_n)$ be?
c) Show that, in every integer multilinear representation, $0 \le f(x_1, \ldots, x_n) \le 1$ whenever $x_1, \ldots, x_n$ are real numbers with $0 \le x_1, \ldots, x_n \le 1$.
d) Similarly, if $f(x_1, \ldots, x_n) \le g(x_1, \ldots, x_n)$ whenever $\{x_1, \ldots, x_n\} \subseteq \{0, 1\}$, then $f(x_1, \ldots, x_n) \le g(x_1, \ldots, x_n)$ whenever $\{x_1, \ldots, x_n\} \subseteq [0\,.\,.\,1]$.
e) If f is monotone and $0 \le x_j \le y_j \le 1$ for $1 \le j \le n$, prove that $f(x) \le f(y)$.

▶ **13.** [*20*] Consider a system that consists of n units, each of which may be "working" or "failing." If x_j represents the condition "unit j is working," then a Boolean function like $x_1 \wedge (\bar{x}_2 \vee \bar{x}_3)$ represents the statement "unit 1 is working, but either unit 2 or unit 3 is failing"; and $S_3(x_1, \ldots, x_n)$ means "exactly three units are working."

Suppose each unit j is in working order with probability p_j, independent of the other units. Show that the Boolean function $f(x_1, \ldots, x_n)$ is true with probability $F(p_1, \ldots, p_n)$, where F is a polynomial in the variables $p_1, \ldots, p_n$.

14. [*20*] The probability function $F(p_1, \ldots, p_n)$ in exercise 13 is often called the *availability* of the system. Find the self-dual function $f(x_1, x_2, x_3)$ of maximum availability when the probabilities (p_1, p_2, p_3) are (a) $(.9, .8, .7)$; (b) $(.8, .6, .4)$; (c) $(.8, .6, .1)$.

▶ **15.** [*M20*] If $f(x_1,\ldots,x_n)$ is any Boolean function, show that there is a polynomial $F(x)$ with the property that $F(x)$ is an integer when x is an integer, and $f(x_1,\ldots,x_n) = F((x_n\ldots x_1)_2) \bmod 2$. *Hint:* Consider $\binom{x}{k} \bmod 2$.

16. [*13*] Can we replace each $\vee$ by $\oplus$ in a full disjunctive normal form?

17. [*10*] By De Morgan's laws, a general disjunctive normal form such as (25) is not only an OR of ANDs, it is a NAND of NANDs:

$$\overline{\overline{(u_{11}\wedge\cdots\wedge u_{1s_1})}\ \wedge\cdots\wedge\ \overline{(u_{m1}\wedge\cdots\wedge u_{ms_m})}}.$$

Both levels of logic can therefore be considered to be identical.

A student named J. H. Quick rewrote this expression in the form

$$(u_{11}\,\bar\wedge\cdots\bar\wedge\, u_{1s_1})\ \bar\wedge\cdots\bar\wedge\ (u_{m1}\,\bar\wedge\cdots\bar\wedge\, u_{ms_m}).$$

Was that a good idea?

▶ **18.** [*20*] Let $u_1\wedge\cdots\wedge u_s$ be an implicant in a disjunctive normal form for a Boolean function f, and let $v_1\vee\cdots\vee v_t$ be a clause in a conjunctive normal form for the same function. Prove that $u_i = v_j$ for some i and j.

19. [*20*] What is the conjunctive prime form of the "random" function in (22)?

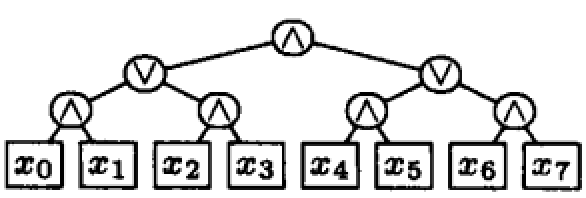
20. [*M21*] True or false: Every prime implicant of $f\wedge g$ can be written $f'\wedge g'$, where f' is a prime implicant of f and g' is a prime implicant of g.

21. [*M20*] Prove that a nonconstant Boolean function is monotone if and only if it can be expressed entirely in terms of the operations $\wedge$ and $\vee$.

22. [*20*] Suppose $f(x_1,\ldots,x_n) = g(x_1,\ldots,x_{n-1}) \oplus h(x_1,\ldots,x_{n-1})\wedge x_n$ as in (16). What conditions on the functions g and h are necessary and sufficient for f to be monotone?

23. [*15*] What is the conjunctive prime form of $(v\wedge w\wedge x)\vee(v\wedge x\wedge z)\vee(x\wedge y\wedge z)$?

24. [*M20*] Consider the complete binary tree with 2^k leaves, illustrated here for $k=3$. Operate alternately with $\wedge$ or $\vee$ on each level, using $\wedge$ at the root, obtaining for example $((x_0\wedge x_1)\vee(x_2\wedge x_3))\wedge((x_4\wedge x_5)\vee(x_6\wedge x_7))$. How many prime implicants does the resulting function contain?



25. [*M21*] How many prime implicants does $(x_1\vee x_2)\wedge(x_2\vee x_3)\wedge\cdots\wedge(x_{n-1}\vee x_n)$ have?

26. [*M23*] Let $\mathcal{F}$ and $\mathcal{G}$ be the families of index sets for the prime clauses and the prime implicants of a monotone CNF and a monotone DNF:

$$f(x) = \bigwedge_{I\in\mathcal{F}}\bigvee_{i\in I} x_i; \qquad g(x) = \bigvee_{J\in\mathcal{G}}\bigwedge_{j\in J} x_j.$$

Exhibit an x such that $f(x)\ne g(x)$ if any of the following conditions hold:

a) There is an $I\in\mathcal{F}$ and a $J\in\mathcal{G}$ with $I\cap J=\emptyset$.

b) $\bigcup_{I\in\mathcal{F}} I \ne \bigcup_{J\in\mathcal{G}} J$.

c) There's an $I\in\mathcal{F}$ with $|I|>|\mathcal{G}|$, or a $J\in\mathcal{G}$ with $|J|>|\mathcal{F}|$.

d) $\sum_{I\in\mathcal{F}} 2^{n-|I|} + \sum_{J\in\mathcal{G}} 2^{n-|J|} < 2^n$, where $n = \bigl|\bigcup_{I\in\mathcal{F}} I\bigr|$.

27. [*M31*] Continuing the previous exercise, consider the following algorithm $\mathrm{X}(\mathcal{F},\mathcal{G})$, which either returns a vector x with $f(x)\ne g(x)$, or returns Λ if $f=g$:

X1. [Check necessary conditions.] Return an appropriate value x if condition (a), (b), (c), or (d) in exercise 26 applies.

X2. [Done?] If $|\mathcal{F}||\mathcal{G}| \le 1$, return Λ.

X3. [Recurse.] Compute the following reduced families, for a "best" index k:

$$\mathcal{F}_1 = \{I \mid I \in \mathcal{F},\ k \notin I\}, \qquad \mathcal{F}_0 = \mathcal{F}_1 \cup \{I \mid k \notin I,\ I \cup \{k\} \in \mathcal{F}\};$$
$$\mathcal{G}_0 = \{J \mid J \in \mathcal{G},\ k \notin J\}, \qquad \mathcal{G}_1 = \mathcal{G}_0 \cup \{J \mid k \notin J,\ J \cup \{k\} \in \mathcal{G}\}.$$

Delete any member of $\mathcal{F}_0$ or $\mathcal{G}_1$ that contains another member of the same family. The index k should be chosen so that the ratio $\rho = \min(|\mathcal{F}_1|/|\mathcal{F}|, |\mathcal{G}_0|/|\mathcal{G}|)$ is as small as possible. If $X(\mathcal{F}_0, \mathcal{G}_0)$ returns a vector x, return the same vector extended with $x_k = 0$. Otherwise if $X(\mathcal{F}_1, \mathcal{G}_1)$ returns a vector x, return the same vector extended with $x_k = 1$. Otherwise return Λ. ▌

If $N = |\mathcal{F}| + |\mathcal{G}|$, prove that step X1 is executed at most $N^{O(\log N)^2}$ times. *Hint:* Show that we always have $\rho \le 1 - 1/\lg N$ in step X3.

28. [*21*] (W. V. Quine, 1952.) If $f(x_1, \ldots, x_n)$ is a Boolean function with prime implicants $p_1, \ldots, p_q$, let $g(y_1, \ldots, y_q) = \bigwedge_{f(x)=1} \bigvee\{y_j \mid p_j(x) = 1\}$. For example, the "random" function (22) is true at the eight points (28), and it has five prime implicants given by (29) and (30); so $g(y_1, \ldots, y_5)$ is

$$(y_1 \vee y_2) \wedge (y_1) \wedge (y_2 \vee y_3) \wedge (y_4) \wedge (y_3 \vee y_5) \wedge (y_5) \wedge (y_5) \wedge (y_4 \vee y_5)$$
$$= (y_1 \wedge y_2 \wedge y_4 \wedge y_5) \vee (y_1 \wedge y_3 \wedge y_4 \wedge y_5)$$

in this case. Prove that every shortest DNF expression for f corresponds to a prime implicant of the monotone function g.

29. [*22*] (The next several exercises are devoted to algorithms that deal with the implicants of Boolean functions by representing points of the n-cube as n-bit numbers $(b_{n-1} \ldots b_1 b_0)_2$, rather than as bit strings $x_1 \ldots x_n$.) Given a bit position j, and given n-bit values $v_0 < v_1 < \cdots < v_{m-1}$, explain how to find all pairs (k, k') such that $0 \le k < k' < m$ and $v_{k'} = v_k \oplus 2^j$, in increasing order of k. The running time of your procedure should be $O(m)$, if bitwise operations on n-bit words take constant time.

▸ **30.** [*27*] The text points out that an implicant of a Boolean function can be regarded as a subcube such as 01∗0∗, contained in the set V of all points for which the function is true. Every subcube can be represented as a pair of binary numbers $a = (a_{n-1} \ldots a_0)_2$ and $b = (b_{n-1} \ldots b_0)_2$, where a records the positions of the asterisks and b records the bits in non-∗ positions. For example, the numbers $a = (00101)_2$ and $b = (01000)_2$ represent the subcube $c = 01{*}0{*}$. We always have $a \mathbin{\&} b = 0$.

The "j-buddy" of a subcube is defined whenever $a_j = 0$, by changing b to $b \oplus 2^j$. For example, 01∗0∗ has three buddies, namely its 4-buddy 11∗0∗, its 3-buddy 00∗0∗, and its 1-buddy 01∗1∗. Every subcube $c \subseteq V$ can be assigned a tag value $(t_{n-1} \ldots t_0)_2$, where $t_j = 1$ if and only if the j-buddy of c is defined and contained in V. With this definition, c represents a maximal subcube (hence a prime implicant) if and only if its tag is zero.

Use these concepts to design an algorithm that finds all maximal subcubes (a, b) of a given set V, where V is represented by the n-bit numbers $v_0 < v_1 < \cdots < v_{m-1}$.

▸ **31.** [*28*] The algorithm in exercise 30 requires a complete list of all points where a Boolean function is true, and that list may be quite long. Therefore we may prefer to work directly with subcubes, never going down to the level of explicit n-tuples unless

necessary. The key to such higher-level methods is the notion of *consensus* between subcubes c and c', denoted by $c \sqcup c'$ and defined to be the largest subcube c'' such that

$$c'' \subseteq c \cup c', \qquad c'' \not\subseteq c, \quad \text{and} \quad c'' \not\subseteq c'.$$

Such a c'' does not always exist. For example, if $c = 000*$ and $c' = *111$, every subcube contained in $c \cup c'$ is contained either in c or in c'.

a) Prove that the consensus, when it exists, can be computed componentwise using the following formulas in each coordinate position:

$$x \sqcup x = x \sqcup * = * \sqcup x = x \quad \text{and} \quad x \sqcup \bar{x} = * \sqcup * = *, \qquad \text{for } x = 0 \text{ and } x = 1.$$

Furthermore, $c \sqcup c'$ exists if and only if the rule $x \sqcup \bar{x} = *$ has been used in exactly one component.

b) A subcube with k asterisks is called a k-cube. Show that, if c is a k-cube and c' is a k'-cube, and if the consensus $c'' = c \sqcup c'$ exists, then c'' is a k''-cube where $1 \le k'' \le \min(k, k') + 1$.

c) If C and C' are families of subcubes, let

$$C \sqcup C' = \{c \sqcup c' \mid c \in C,\ c' \in C', \text{ and } c \sqcup c' \text{ exists}\}.$$

Explain why the following algorithm works.

Algorithm E (*Find maximal subcubes*). Given a family C of subcubes of the n-cube, this algorithm outputs the maximal subcubes of $V = \bigcup_{c \in C} c$, without actually computing the set V itself.

E1. [Initialize.] Set $j \leftarrow 0$. Delete any subcube c of C that is contained in another.

E2. [Done?] (At this point, every j-cube $\subseteq V$ is contained in some element of C, and C contains no k-cubes with $k < j$.) If C is empty, the algorithm terminates.

E3. [Take consensuses.] Set $C' \leftarrow C \sqcup C$, and remove all subcubes from C' that are k-cubes for $k \le j$. While performing this computation, also output any j-cube $c \in C$ for which $c \sqcup C$ does not produce a $(j+1)$-cube of C'.

E4. [Advance.] Set $C \leftarrow C \cup C'$, but delete all j-cubes from this union. Then delete any subcube $c \in C$ that is contained in another. Set $j \leftarrow j+1$ and go to E2. ▮

(See exercise 7.1.3–142 for an efficient way to perform these computations.)

▸ **32.** [*M29*] Let $c_1, \ldots, c_m$ be subcubes of the n-cube.

a) Prove that $c_1 \cup \cdots \cup c_m$ contains at most one maximal subcube c that is not contained in $c_1 \cup \cdots \cup c_{j-1} \cup c_{j+1} \cup \cdots \cup c_m$ for any $j \in \{1, \ldots, m\}$. (If c exists, we call it the *generalized consensus* of $c_1, \ldots, c_m$, because $c = c_1 \sqcup c_2$ in the notation of exercise 31 when $m = 2$.)

b) Find a set of m subcubes for which each of the $2^m - 1$ nonempty subsets of $\{c_1, \ldots, c_m\}$ has a generalized consensus.

c) Prove that a DNF with m implicants has at most $2^m - 1$ prime implicants.

d) Find a DNF that has m implicants and $2^m - 1$ prime implicants.

33. [*M21*] Let $f(x_1, \ldots, x_n)$ be one of the $\binom{2^n}{m}$ Boolean functions that are true at exactly m points. If f is chosen at random, what is the probability that $x_1 \wedge \cdots \wedge x_k$ is (a) an implicant of f? (b) a prime implicant of f? [Give the answer to part (b) as a sum; but evaluate it in closed form when $k = n$.]

▶ **34.** [*HM37*] Continuing exercise 33, let $c(m,n)$ be the average total number of implicants, and let $p(m,n)$ be the average total number of prime implicants.

a) If $0 \le m \le 2^n/n$, show that $m \le c(m,n) \le \frac{3}{2}m + O(m/n)$ and $p(m,n) \ge me^{-1} + O(m/n)$; hence $p(m,n) = \Theta(c(m,n))$ in this range.

b) Now let $2^n/n \le m \le (1-\epsilon)2^n$, where ϵ is a fixed positive constant. Define the numbers t and α_{mn} by the relations

$$n^{-4/3} \le \left(\frac{m}{2^n}\right)^{2^t} = \alpha_{mn} < n^{-2/3}, \qquad \text{integer } t.$$

Express the asymptotic values of $c(m,n)$ and $p(m,n)$ in terms of n, t, and α_{mn}. [*Hint:* Show that almost all of the implicants have exactly $n-t$ or $n-t-1$ literals.]

c) Estimate $c(m,n)/p(m,n)$ when $m = 2^{n-1}$ and $n = \lfloor(\ln t - \ln\ln t)2^{2^t}\rfloor$, integer t.

d) Prove that $c(m,n)/p(m,n) = O(\log\log n/\log\log\log n)$ when $m \le (1-\epsilon)2^n$.

▶ **35.** [*M25*] A DNF is called *orthogonal* if its implicants correspond to disjoint subcubes. Orthogonal disjunctive normal forms are particularly useful when the reliability polynomial of exercise 13 is being calculated or estimated.

The full DNF of every function is obviously orthogonal, because its subcubes are single points. But we can often find an orthogonal DNF that has significantly fewer implicants, especially when the function is monotone. For example, the function $(x_1\wedge x_2)\vee(x_2\wedge x_3)\vee(x_3\wedge x_4)$ is true at eight points, and it has the orthogonal DNF

$$(x_1\wedge x_2)\vee(\bar{x}_1\wedge x_2\wedge x_3)\vee(\bar{x}_2\wedge x_3\wedge x_4).$$

In other words, the overlapping subcubes 11**, *11*, **11 can be replaced by the disjoint subcubes 11**, 011*, *011. Using the binary notation for subcubes in exercise 30, these subcubes have asterisk codes 0011, 0001, 1000 and bit codes 1100, 0110, 0011.

Every monotone function can be defined by a list of bit codes $B_1, \ldots, B_p$, when the asterisk codes are respectively $\bar{B}_1, \ldots, \bar{B}_p$. Given such a list, let the "shadow" S_k of B_k be the bitwise OR of $B_j \,\&\, \bar{B}_k$, for all $1 \le j < k$ such that $\nu(B_j \,\&\, \bar{B}_k) = 1$:

$$S_k = \beta_{1k} \mid \cdots \mid \beta_{(k-1)k}, \quad \beta_{jk} = ((B_j\&\bar{B}_k) \oplus ((B_j\&\bar{B}_k)-1)) \dot{-} ((B_j\&\bar{B}_k)-1).$$

For example, when the bit codes are $(B_1,B_2,B_3) = (1100, 0110, 0011)$, we get the shadow codes $(S_1,S_2,S_3) = (0000, 1000, 0100)$.

a) Show that the asterisk codes $A'_j = \bar{B}_j - S_j$ and bit codes B_j define subcubes that cover the same points as the subcubes with asterisk codes $A_j = \bar{B}_j$.

b) A list of bit codes $B_1, \ldots, B_p$ is called a *shelling* if $B_j \,\&\, S_k$ is nonzero for all $1 \le j < k \le p$. For example, (1100, 0110, 0011) is a shelling; but if we arrange those bit codes in the order (1100, 0011, 0110) the shelling condition fails when $j = 1$ and $k = 2$, although we do have $S_3 = 1001$. Prove that the subcubes in part (a) are disjoint if and only if the list of bit codes is a shelling.

c) According to Theorem Q, every prime implicant must appear among the B's when we represent a monotone Boolean function in this way. But sometimes we need to add additional implicants if we want the subcubes to be disjoint. For example, there is no shelling for the bit codes 1100 and 0011. Show that we can, however, obtain a shelling for this function $(x_1\wedge x_2)\vee(x_3\wedge x_4)$ by adding one more bit code. What is the resulting orthogonal DNF?

d) Permute the bit codes {11000, 01100, 00110, 00011, 11010} to obtain a shelling.

e) Add two bit codes to the set {110000, 011000, 001100, 000110, 000011} in order to make a shellable list.

36. [*M21*] Continuing exercise 35, let f be any monotone function, not identically 1. Show that the set of bit vectors

$$B = \{\, x \mid f(x) = 1 \text{ and } f(x') = 0 \,\}, \qquad x' = x \,\&\, (x-1),$$

is always shellable when listed in decreasing lexicographic order. (The vector x' is obtained from x by zeroing out the rightmost 1.) For example, this method produces an orthogonal DNF for $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ from the list (1100, 1011, 0111, 0011).

▶ **37.** [*M31*] Find a shellable DNF for $(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \cdots \vee (x_{2n-1} \wedge x_{2n})$ that has $2^n - 1$ implicants, and prove that no orthogonal DNF for this function has fewer.

38. [*05*] Is it hard to test the satisfiability of functions in *disjunctive* normal form?

▶ **39.** [*25*] Let $f(x_1, \ldots, x_n)$ be a Boolean formula represented as an extended binary tree with N internal nodes and $N + 1$ leaves. Each leaf is labeled with a variable x_k, and each internal node is labeled with one of the sixteen binary operators in Table 1; applying the operators from bottom to top yields $f(x_1, \ldots, x_n)$ as the value of the root.

Explain how to construct a formula $F(x_1, \ldots, x_n, y_1, \ldots, y_N)$ in 3CNF, having exactly $4N + 1$ clauses, such that $f(x_1, \ldots, x_n) = \exists y_1 \ldots \exists y_N F(x_1, \ldots, x_n, y_1, \ldots, y_N)$. (Thus f is satisfiable if and only if F is satisfiable.)

40. [*23*] Given an undirected graph G, construct the following clauses on the Boolean variables $\{p_{uv} \mid u \neq v\} \cup \{q_{uvw} \mid u \neq v,\ u \neq w,\ v \neq w,\ u \not- w\}$, where u, v, and w denote vertices of G:

$$A = \bigwedge\{\, (p_{uv} \vee p_{vu}) \wedge (\bar p_{uv} \vee \bar p_{vu}) \mid u \neq v \,\};$$
$$B = \bigwedge\{\, (\bar p_{uv} \vee \bar p_{vw} \vee p_{uw}) \mid u \neq v,\ u \neq w,\ v \neq w \,\};$$
$$C = \bigwedge\{\, (\bar q_{uvw} \vee p_{uv}) \wedge (\bar q_{uvw} \vee p_{vw}) \wedge (q_{uvw} \vee \bar p_{uv} \vee \bar p_{vw}) \mid u \neq v,\ u \neq w,\ v \neq w,\ u \not- w \,\};$$
$$D = \bigwedge\{ (\textstyle\bigvee_{v \notin \{u,w\}} (q_{uvw} \vee q_{wvu})) \mid u \neq w,\ u \not- w \,\}.$$

Prove that the formula $A \wedge B \wedge C \wedge D$ is satisfiable if and only if G has a Hamiltonian path. *Hint:* Think of p_{uv} as the statement '$u < v$'.

41. [*20*] (*The pigeonhole principle.*) The island of San Serriffe contains m pigeons and n holes. Find a conjunctive normal form that is satisfiable if and only if each pigeon can be the sole occupant of at least one hole.

42. [*20*] Find a short, unsatisfiable CNF that is not totally trivial, although it consists entirely of Horn clauses that are also Krom clauses.

43. [*20*] Is there an efficient way to decide satisfiability of a conjunctive normal form that consists entirely of Horn clauses and/or Krom clauses (possibly mixed)?

44. [*M23*] Complete the proof of Theorem H by studying the implications of (33).

45. [*M20*] (a) Show that exactly half of the Horn functions of n variables are definite. (b) Also show that there are more Horn functions of n variables than monotone functions of n variables (unless $n = 0$).

46. [*20*] Which of the 11×11 character pairs **xy** can occur next to each other in the context-free grammar (34)?

47. [*20*] Given a sequence of relations $j \prec k$ with $1 \le j, k \le n$ as in Algorithm 2.2.3T (topological sorting), consider the clauses

$$x_{j_1} \wedge \cdots \wedge x_{j_t} \Rightarrow x_k \qquad \text{for } 1 \le k \le n,$$

where $\{j_1, \ldots, j_t\}$ is the set of elements such that $j_i \prec k$. Compare the behavior of Algorithm C on these clauses to the behavior of Algorithm 2.2.3T.

▸ **48.** [*21*] What's a good way to test a set of Horn clauses for satisfiability?

49. [*22*] Show that, if $f(x_1,\ldots,x_n)$ and $g(x_1,\ldots,x_n)$ are both defined by Horn clauses in CNF, there is an easy way to test if $f(x_1,\ldots,x_n) \le g(x_1,\ldots,x_n)$ for all $x_1, \ldots, x_n$.

50. [*HM42*] There are $(n+2)2^{n-1}$ possible Horn clauses on n variables. Select $c \cdot 2^n$ of them at random, with repetition permitted, where $c > 0$; and let $P_n(c)$ be the probability that all of the selected clauses are simultaneously satisfiable. Prove that

$$\lim_{n\to\infty} P_n(c) \;=\; 1-(1-e^{-c})(1-e^{-2c})(1-e^{-4c})(1-e^{-8c})\ldots.$$

▸ **51.** [*22*] A great many two-player games can be defined by specifying a directed graph in which each vertex represents a game position. There are two players, Alice and Bob, who construct an oriented path by starting at a particular vertex and taking turns to extend the path, one arc at a time. Before the game starts, each vertex has either been marked A (meaning that Alice wins), or marked B (meaning that Bob wins), or marked C (meaning that the cat wins), or left unmarked.

When the path reaches a vertex v marked A or B, that player wins. The game stops without a winner if v has been visited before, with the same player to move. If v is marked C, the currently active player has the option of accepting a draw; otherwise he or she must choose an outgoing arc to extend the path, and the other player becomes active. (If v is an unmarked vertex with out-degree zero, the active player loses.)

Associating four propositional variables $A^+(v)$, $A^-(v)$, $B^+(v)$, and $B^-(v)$ with every vertex v of the graph, explain how to construct a set of definite Horn clauses such that $A^+(v)$ is in the core if and only if Alice can force a win when the path starts at v and she moves first; $A^-(v)$ is in the core if and only if Bob can force her to lose in that game; $B^+(v)$ and $B^-(v)$ are similar to $A^+(v)$ and $A^-(v)$, but with roles reversed.

52. [*25*] (*Boolean games.*) Any Boolean function $f(x_1,\ldots,x_n)$ leads to a game called "two steps forward or one step back," in the following way: There are two players, 0 and 1, who repeatedly assign values to the variables x_j; player y tries to make $f(x_1,\ldots,x_n)$ equal to y. Initially all variables are unassigned, and the position marker m is zero. Players take turns, and the currently active player either sets $m \leftarrow m+2$ (if $m+2 \le n$) or $m \leftarrow m-1$ (if $m-1 \ge 1$), then sets

$$\begin{cases} x_m \leftarrow 0 \text{ or } 1, & \text{if } x_m \text{ was not previously assigned;} \\ x_m \leftarrow \bar{x}_m, & \text{if } x_m \text{ was previously assigned.} \end{cases}$$

The game is over as soon as a value has been assigned to all variables; then $f(x_1,\ldots,x_n)$ is the winner. A draw is declared if the same state (including the value of m) is reached twice. Notice that at most four moves are possible at any time.

Study examples of this game when $2 \le n \le 9$, in the following four cases:

a) $f(x_1,\ldots,x_n) = [x_1\ldots x_n < x_n \ldots x_1]$ (in lexicographic order);
b) $f(x_1,\ldots,x_n) = x_1 \oplus \cdots \oplus x_n$;
c) $f(x_1,\ldots,x_n) = [x_1\ldots x_n$ contains no two consecutive 1s];
d) $f(x_1,\ldots,x_n) = [(x_1\ldots x_n)_2$ is prime].

53. [*23*] Show that the impossible comedy festival of (37) *can* be scheduled if a change is made to the requirements of only (a) Tomlin; (b) Unwin; (c) Vegas; (d) Xie; (e) Yankovic; (f) Zany.

54. [*20*] Let $S = \{u_1, u_2, \ldots, u_k\}$ be the set of literals in some strong component of a digraph that corresponds to a 2CNF formula as in Fig. 6. Show that S contains both a variable and its complement if and only if $u_j = \bar{u}_1$ for some j with $2 \le j \le k$.

▶ **55.** [*30*] Call $f(x_1,\ldots,x_n)$ a *renamed Horn function* if there are Boolean constants $y_1,\ldots,y_n$ such that $f(x_1\oplus y_1,\ldots,x_n\oplus y_n)$ is a Horn function.

a) Given $f(x_1,\ldots,x_n)$ in CNF, explain how to construct $g(y_1,\ldots,y_n)$ in 2CNF so that the clauses of $f(x_1\oplus y_1,\ldots,x_n\oplus y_n)$ are Horn clauses if and only if $g(y_1,\ldots,y_n)=1$.

b) Design an algorithm that decides in $O(m)$ steps whether or not all clauses of a given CNF of length m can be converted into Horn clauses by complementing some subset of the variables.

▶ **56.** [*20*] The satisfiability problem for a Boolean function $f(x_1,x_2,\ldots,x_n)$ can be stated formally as the question of whether or not the quantified formula

$$\exists x_1\,\exists x_2\ldots\exists x_n\,f(x_1,x_2,\ldots,x_n)$$

is true; here '$\exists x_j\,\alpha$' means, "there exists a Boolean value x_j such that α holds."

A much more general evaluation problem arises when we replace one or more of the existential quantifiers $\exists x_j$ by the universal quantifier $\forall x_j$, where '$\forall x_j\,\alpha$' means, "for all Boolean values x_j, α holds."

Which of the eight quantified formulas $\exists x\,\exists y\,\exists z\,f(x,y,z)$, $\exists x\,\exists y\,\forall z\,f(x,y,z)$, …, $\forall x\,\forall y\,\forall z\,f(x,y,z)$ are true when $f(x,y,z)=(x\vee y)\wedge(\bar{x}\vee z)\wedge(y\vee\bar{z})$?

▶ **57.** [*30*] (B. Aspvall, M. F. Plass, and R. E. Tarjan.) Continuing exercise 56, design an algorithm that decides in linear time whether or not a given fully quantified formula $f(x_1,\ldots,x_n)$ is true, when f is any formula in 2CNF (any conjunction of Krom clauses).

▶ **58.** [*37*] Continuing exercise 57, design an efficient algorithm that decides whether or not a given fully quantified conjunction of *Horn* clauses is true.

▶ **59.** [*M20*] (D. Pehoushek and R. Fraer, 1997.) If the truth table for $f(x_1,x_2,\ldots,x_n)$ has a 1 in exactly k places, show that exactly k of the fully quantified formulas $\mathsf{Q}x_1\,\mathsf{Q}x_2\ldots\mathsf{Q}x_n\,f(x_1,x_2,\ldots,x_n)$ are true, when each Q is either $\exists$ or $\forall$.

60. [*12*] Which of the following expressions yield the median $\langle xyz\rangle$, as defined in (43)?
(a) $(x\wedge y)\oplus(y\wedge z)\oplus(x\wedge z)$. (b) $(x\vee y)\oplus(y\vee z)\oplus(x\vee z)$. (c) $(x\oplus y)\wedge(y\oplus z)\wedge(x\oplus z)$.
(d) $(x\equiv y)\oplus(y\equiv z)\oplus(x\equiv z)$. (e) $(x\,\overline{\wedge}\,y)\wedge(y\,\overline{\wedge}\,z)\wedge(x\,\overline{\wedge}\,z)$. (f) $(x\,\overline{\wedge}\,y)\vee(y\,\overline{\wedge}\,z)\vee(x\,\overline{\wedge}\,z)$.

61. [*13*] True or false: If $\circ$ is any one of the Boolean binary operations in Table 1, we have the distributive law $w\circ\langle xyz\rangle=\langle(w\circ x)(w\circ y)(w\circ z)\rangle$.

62. [*25*] (C. Schensted.) If $f(x_1,\ldots,x_n)$ is a monotone Boolean function and $n\geq 3$, prove the median expansion formula

$$f(x_1,\ldots,x_n)=\langle f(x_1,x_1,x_3,x_4,\ldots,x_n)f(x_1,x_2,x_2,x_4,\ldots,x_n)f(x_3,x_2,x_3,x_4,\ldots,x_n)\rangle.$$

63. [*20*] Equation (49) shows how to compute the median of five elements via medians of three. Conversely, can we compute $\langle xyz\rangle$ with a subroutine for medians of five?

64. [*23*] (S. B. Akers, Jr.) (a) Prove that a Boolean function $f(x_1,\ldots,x_n)$ is monotone and self-dual if and only if it satisfies the following condition:

For all $x=x_1\ldots x_n$ and $y=y_1\ldots y_n$ there exists k such that $f(x)=x_k$ and $f(y)=y_k$.

(b) Suppose f is undefined for certain values, but the stated condition holds whenever both $f(x)$ and $f(y)$ are defined. Show that there is a monotone self-dual Boolean function g for which $g(x)=f(x)$ whenever $f(x)$ is defined.

▶ **65.** [*M21*] Any subset X of $\{1,2,\ldots,n\}$ corresponds to a binary vector $x=x_1x_2\ldots x_n$ via the rule $x_j=[j\in X]$. And any family $\mathcal{F}$ of such subsets corresponds to a Boolean function $f(x)=f(x_1,x_2,\ldots,x_n)$ of n variables, via the rule $f(x)=[X\in\mathcal{F}]$. Therefore

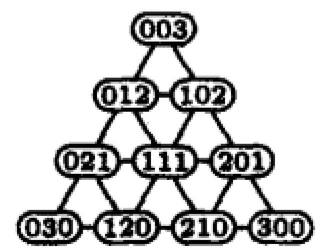
every statement about families of subsets corresponds to a statement about Boolean functions, and vice versa.

A family $\mathcal{F}$ is called *intersecting* if $X \cap Y \neq \emptyset$ whenever $X, Y \in \mathcal{F}$. An intersecting family that loses this property whenever we try to add another subset is said to be *maximal*. Prove that $\mathcal{F}$ is a maximal intersecting family if and only if the corresponding Boolean function f is monotone and self-dual.

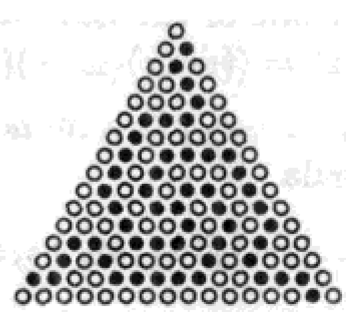
▶ **66.** [*M25*] A *coterie* of $\{1, \ldots, n\}$ is a family $\mathcal{C}$ of subsets called *quorums*, which have the following properties whenever $Q \in \mathcal{C}$ and $Q' \in \mathcal{C}$: (i) $Q \cap Q' \neq \emptyset$; (ii) $Q \subseteq Q'$ implies $Q = Q'$. Coterie $\mathcal{C}$ *dominates* coterie $\mathcal{C}'$ if $\mathcal{C} \neq \mathcal{C}'$ and if, for every $Q' \in \mathcal{C}'$, there is a $Q \in \mathcal{C}$ with $Q \subseteq Q'$. For example, the coterie $\{\{1,2\},\{2,3\}\}$ is dominated by $\{\{1,2\},\{1,3\},\{2,3\}\}$ and also by $\{\{2\}\}$. [Coteries were introduced in classic papers by L. Lamport, *CACM* **21** (1978), 558–565; H. Garcia-Molina and D. Barbara, *JACM* **32** (1985), 841–860. They have numerous applications to distributed system protocols, including mutual exclusion, data replication, and name servers. In these applications $\mathcal{C}$ is preferred to any coterie that it dominates.]

Prove that $\mathcal{C}$ is a nondominated coterie if and only if its quorums are the index sets of variables in the prime implicants of a monotone self-dual Boolean function $f(x_1, \ldots, x_n)$. (Thus Table 2 illustrates the nondominated coteries on $\{1,2,3,4\}$.)

▶ **67.** [*M30*] (J. W. Milnor and C. Schensted.) A triangular grid of order n, illustrated here for $n = 3$, contains $(n+2)(n+1)/2$ points with nonnegative "barycentric coordinates" xyz, where $x+y+z = n$. Two points are adjacent if they differ by ± 1 in exactly two coordinate positions. A point is said to lie on the x side if its x coordinate is zero, on the y side if its y coordinate is zero, or on the z side if its z coordinate is zero; thus each side contains $n+1$ points. If $n > 0$, a point lies on two different sides if and only if it occupies one of the three corner positions.



A "Y" is a connected set of points with at least one point on each side. Suppose each vertex of a triangular grid is covered with a white stone or a black stone. For example, the 52 black stones in



contain a (somewhat distorted) Y; but if any of them is changed from black to white, there is a white Y instead. A moment's thought makes it intuitively clear that, in any placement, the black stones contain a Y if and only if the white stones do not.

We can represent the color of each stone by a Boolean variable, with 0 for white and 1 for black. Let $Y(t) = 1$ if and only if there's a black Y, where t is a triangular grid comprising all the Boolean variables. This function Y is clearly monotone; and the intuitive claim made in the preceding paragraph is equivalent to saying that Y is also self-dual. The purpose of this exercise is to prove the claim rigorously, using median algebra.

Given $a, b, c \geq 0$, let t_{abc} be the triangular subgrid containing all points whose coordinates xyz satisfy $x \geq a$, $y \geq b$, $z \geq c$. For example, t_{001} denotes all points except those on the z side (the bottom row). Notice that, if $a+b+c = n$, t_{abc} is the single point with coordinates abc; and in general, t_{abc} is a triangular grid of order $n-a-b-c$.

a) Prove that, if $n > 0$, $Y(t) = \langle Y(t_{100})Y(t_{010})Y(t_{001})\rangle$.

b) If $n > 0$, let t^* be the triangular grid of order $n-1$ defined by the rule

$$t^*_{xyz} = \langle t_{(x+1)yz}t_{x(y+1)z}t_{xy(z+1)}\rangle, \qquad \text{for } x+y+z=n-1.$$

Prove that $Y(t) = Y(t^*)$. [In other words, t^* condenses each small triangle of stones by taking the median of their colors. Repeating this process defines a *pyramid* of stones, with the top stone black if and only if there is a black Y at the bottom. It's fun to apply this condensation principle to the twisted Y above.]

68. [*46*] The just-barely-Y configuration shown in the previous exercise has 52 black stones. What is the largest number of black stones possible in such a configuration? (That is, how many variables can there be in a prime implicant of the function $Y(t)$?)

▶ **69.** [*M26*] (C. Schensted.) Exercise 67 expresses the Y function in terms of medians. Conversely, let $f(x_1,\ldots,x_n)$ be any monotone self-dual Boolean function with $m+1$ prime implicants p_0, p_1, …, p_m. Prove that $f(x_1,\ldots,x_n) = Y(T)$, where T is any triangular grid of order $m-1$ in which T_{abc} is a variable common to p_a and p_{a+b+1}, for $a+b+c=m-1$. For example, when $f(w,x,y,z) = \langle xwywz\rangle$ we have $m=3$ and

$$f(w,x,y,z) = (w\wedge x)\vee(w\wedge y)\vee(w\wedge z)\vee(x\wedge y\wedge z) = Y\left(\begin{smallmatrix} & w & \\ w & & w \\ x & y & z\end{smallmatrix}\right).$$

▶ **70.** [*M20*] (A. Meyerowitz, 1989.) Given any monotone self-dual Boolean function $f(x) = f(x_1,\ldots,x_n)$, choose any prime implicant $x_{j_1}\wedge\cdots\wedge x_{j_s}$ and let

$$g(x) = \bigl(f(x)\wedge[x\neq t]\bigr)\vee[x=\bar t],$$

where $t = t_1\ldots t_n$ is the bit vector that has 1s in positions $\{j_1,\ldots,j_s\}$. Prove that $g(x)$ is also monotone and self-dual. (Notice that $g(x)$ is equal to $f(x)$ except at the two points t and $\bar t$.)

▶ **71.** [*M21*] Given the axioms (50), (51), and (52) of a median algebra, prove that the long distributive law (54) is a consequence of the shorter law (53).

72. [*M22*] Derive (58), (59), and (60) from the median laws (50)–(53).

73. [*M32*] (S. P. Avann.) Given a median algebra M, whose intervals are defined by (57) and whose corresponding median graph is defined by (61), let $d(u,v)$ denote the distance from u to v. Also let '$[uxv]$' stand for the statement "x lies on a shortest path from u to v."

a) Prove that $[uxv]$ holds if and only if $d(u,v) = d(u,x) + d(x,v)$.

b) Suppose $x\in[u\,..\,v]$ and $u\in[x\,..\,y]$, where $x\neq u$ and y — v is an edge of the graph. Show that x — u is also an edge.

c) If $x\in[u\,..\,v]$, prove $[uxv]$, by induction on $d(u,v)$.

d) Conversely, prove that $[uxv]$ implies $x\in[u\,..\,v]$.

74. [*M21*] In a median algebra, show that $w = \langle xyz\rangle$ whenever we have $w\in[x\,..\,y]$, $w\in[x\,..\,z]$, and $w\in[y\,..\,z]$ according to definition (57).

▶ **75.** [*M36*] (M. Sholander, 1954.) Suppose M is a set of points with a betweenness relation "x lies between u and v," symbolized by $[uxv]$, which satisfies the following three axioms:

i) If $[uvu]$ then $u=v$.

ii) If $[uxv]$ and $[xyu]$ then $[vyu]$.

iii) Given x, y, and z, exactly one point $w = \langle xyz\rangle$ satisfies $[xwy]$, $[xwz]$, and $[ywz]$.

The object of this exercise is to prove that M is a median algebra.

a) Prove the majority law $\langle xxy\rangle = x$, Eq. (50).

b) Prove the commutative law $\langle xyz\rangle = \langle xzy\rangle = \cdots = \langle zyx\rangle$, Eq. (51).
c) Prove that $[uxv]$ if and only if $x = \langle uxv\rangle$.
d) If $[uxy]$ and $[uyv]$, prove that $[xyv]$.
e) If $[uxv]$ and $[uyz]$ and $[vyz]$, prove that $[xyz]$. *Hint:* Construct the points $w = \langle yuv\rangle$, $p = \langle wux\rangle$, $q = \langle wvx\rangle$, $r = \langle pxz\rangle$, $s = \langle qxz\rangle$, and $t = \langle rsz\rangle$.
f) Finally, deduce the short distributive law, Eq. (53): $\langle\langle xyz\rangle uv\rangle = \langle x\langle yuv\rangle\langle zuv\rangle\rangle$.

76. [*M33*] Derive the betweenness axioms (i), (ii), and (iii) of exercise 75, starting from the three median axioms (50), (51), and (52), letting $[uxv]$ be an abbreviation for "$x = \langle uxv\rangle$." Do not use the distributive law (53). *Hint:* See exercise 74.

77. [*M28*] Let G be a median graph containing the edge r — s. For each edge u — v, call u an *early neighbor* of v if and only if r is closer to u than to v. Partition the vertices into "left" and "right" parts, where left vertices are closer to r than to s and right vertices are closer to s than to r. Each right vertex v has a *rank*, which is the shortest distance from v to a left vertex. Similarly, each left vertex u has rank $1 - d$, where d is the shortest distance from u to a right vertex. Thus u has rank zero if it is adjacent to a right vertex, otherwise its rank is negative. Vertex r clearly has rank 0, and s has rank 1.
a) Show that every vertex of rank 1 is adjacent to exactly one vertex of rank 0.
b) Show that the set of all right vertices is convex.
c) Show that the set of all vertices with rank 1 is convex.
d) Prove that steps I3–I9 of Subroutine I correctly mark all vertices of ranks 1 and 2.
e) Prove that Algorithm H is correct.

▶ **78.** [*M26*] If the vertex v is examined k times in step I4 during the execution of Algorithm H, prove that the graph has at least 2^k vertices. *Hint:* There are k ways to start a shortest path from v to a; thus at least k 1s appear in $l(v)$.

▶ **79.** [*M27*] (R. L. Graham.) A *subgraph of a hypercube* is a graph whose vertices v can be labeled with bit strings $l(v)$ in such a way that u — v if and only if $l(u)$ and $l(v)$ differ in exactly one bit position. (Each label has the same length.)
a) One way to define an n-vertex subgraph of a hypercube is to let $l(v)$ be the binary representation of v, for $0 \le v < n$. Show that this subgraph has exactly $f(n) = \sum_{k=0}^{n-1} \nu(k)$ edges, where $\nu(k)$ is the sideways addition function.
b) Prove that $f(n) \le n\lceil \lg n\rceil/2$.
c) Prove that no n-vertex subgraph of a hypercube has more than $f(n)$ edges.

80. [*27*] A *partial cube* is an "isometric" subgraph of a hypercube, namely a subgraph in which the distances between vertices are the same as they are in the full graph. The vertices of a partial cube can therefore be labeled in such a way that the distance from u to v is the "Hamming distance" between $l(u)$ and $l(v)$, namely $\nu(l(u) \oplus l(v))$. Algorithm H shows that every median graph is a partial cube.
a) Find a subgraph of the 4-cube that isn't a partial cube.
b) Give an example of a partial cube that isn't a median graph.

81. [*16*] Is every median graph bipartite?

82. [*25*] (*Incremental changes in service.*) Given a sequence of vertices $(v_0, v_1, \ldots, v_t)$ in a graph G, consider the problem of finding another sequence $(u_0, u_1, \ldots, u_t)$ for which $u_0 = v_0$ and the sum

$$(d(u_0,u_1) + d(u_1,u_2) + \cdots + d(u_{t-1},u_t)) \;+\; (d(u_1,v_1) + d(u_2,v_2) + \cdots + d(u_t,v_t))$$

is minimized, where $d(u,v)$ denotes the distance from u to v. (Each v_k can be regarded as a request for a resource needed at that vertex; a server moves to u_k as those requests are handled in sequence.) Prove that if G is a median graph, we get an optimum solution by choosing $u_k = \langle u_{k-1}v_kv_{k+1}\rangle$ for $0 < k < t$, and $u_t = v_t$.

▶ **83.** [*28*] Generalizing exercise 82, find an efficient way to minimize

$$(d(u_0,u_1)+d(u_1,u_2)+\cdots+d(u_{t-1},u_t)) \;+\; \rho(d(u_1,v_1)+d(u_2,v_2)+\cdots+d(u_t,v_t))$$

in a median graph, given any positive ratio ρ.

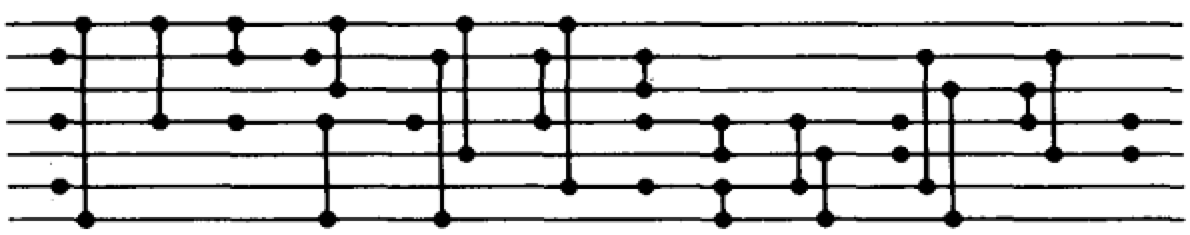
84. [*30*] Write a program to find all monotone self-dual Boolean functions of five variables. What are the edges of the corresponding median graph? (Table 2 illustrates the four-variable case.)

▶ **85.** [*M22*] Theorem S tells us that every formula in 2CNF corresponds to a median set; therefore every antisymmetric digraph such as Fig. 6 also corresponds to a median set. Precisely which of those digraphs correspond to *reduced* median sets?

86. [*15*] If v, w, x, y, and z belong to a median set X, does their five-element median $\langle vwxyz\rangle$, computed componentwise, always belong to X?

87. [*24*] What CI-net does the proof of Theorem F construct for the free tree (63)?

88. [*M21*] We can use parallel computation to condense the network (74) into



by letting each module act at the earliest possible time. Prove that, although the network constructed in the proof of Theorem F may contain $\Omega(t^2)$ modules, it always requires at most $O(t\log t)$ levels of delay.

89. [*24*] When the construction (73) appends a new cluster of modules to enforce the condition $u \to v$, for some literals u and v, prove that it preserves all previously enforced conditions $u' \to v'$.

▶ **90.** [*21*] Construct a CI-net with input bits $x_1 \ldots x_t$ and output bits $y_1 \ldots y_t$, where $y_1 = \cdots = y_{t-1} = 0$ and $y_t = x_1 \oplus \cdots \oplus x_t$. Try for only $O(\log t)$ levels of delay.

91. [*46*] Can a retraction mapping for the labels of every median graph of dimension t be computed by a CI-net that has only $O(\log t)$ levels of delay? [This question is motivated by the existence of asymptotically optimum networks for the analogous problem of sorting; see M. Ajtai, J. Komlós, and E. Szemerédi, *Combinatorica* **3** (1983), 1–19.]

92. [*46*] Can a CI-net sort n Boolean inputs with fewer modules than a "pure" sorting network that has no inverters?

93. [*M20*] Prove that every retract X of a graph G is an isometric subgraph of G. (In other words, distances in X are the same as in G; see exercise 80.)

94. [*M21*] Prove that every retract X of a hypercube is a set of median labels, if we suppress coordinates that are constant for all $x \in X$.

95. [*M25*] True or false: The set of all outputs produced by a comparator-inverter network, when the inputs range over all possible bit strings, is always a median set.

96. [*HM25*] Instead of insisting that the constants $w_1, w_2, \ldots, w_n$, and t in (75) must be integers, we could allow them to be arbitrary real numbers. Would that increase the number of threshold functions?

97. [*10*] What median/majority functions arise in (8_1) when $n = 2$, $w_1 = w_2 = 1$, and $t = -1$, 0, 1, 2, 3, or 4?

98. [*M23*] Prove that any self-dual threshold function can be expressed in the form

$$f(x_1, x_2, \ldots, x_n) = [v_1 y_1 + \cdots + v_n y_n > 0],$$

where each y_j is either x_j or $\bar{x}_j$. For example, $2x_1 + 3x_2 + 5x_3 + 7x_4 + 11x_5 + 13x_6 \geq 21$ if and only if $2x_1 + 3x_2 + 5x_3 - 7\bar{x}_4 + 11x_5 - 13\bar{x}_6 > 0$.

▶ **99.** [*20*] (J. E. Mezei, 1961.) Prove that

$$\langle\langle x_1 \ldots x_{2s-1}\rangle y_1 \ldots y_{2t-2}\rangle = \langle x_1 \ldots x_{2s-1} y_1^s \ldots y_{2t-2}^s\rangle.$$

100. [*20*] True or false: If $f(x_1, \ldots, x_n)$ is a threshold function, so are the functions $f(x_1, \ldots, x_n) \wedge x_{n+1}$ and $f(x_1, \ldots, x_n) \vee x_{n+1}$.

101. [*M23*] The *Fibonacci threshold function* $F_n(x_1, \ldots, x_n)$ is defined by the formula $\langle x_1^{F_1} x_2^{F_2} \ldots x_{n-1}^{F_{n-1}} x_n^{F_{n-2}}\rangle$ when $n \geq 3$; for example, $F_7(x_1, \ldots, x_7) = \langle x_1 x_2 x_3^2 x_4^3 x_5^5 x_6^8 x_7^5\rangle$.

a) What are the prime implicants of $F_n(x_1, \ldots, x_n)$?

b) Find an orthogonal DNF for $F_n(x_1, \ldots, x_n)$ (see exercise 35).

c) Express $F_n(x_1, \ldots, x_n)$ in terms of the Y function (see exercises 67 and 69).

102. [*M21*] The *self-dualization* of a Boolean function is defined by the formulas

$$\begin{aligned}\hat{f}(x_0, x_1, \ldots, x_n) &= (x_0 \wedge f(x_1, \ldots, x_n)) \vee (\bar{x}_0 \wedge \overline{f(\bar{x}_1, \ldots, \bar{x}_n)}) \\ &= (\bar{x}_0 \vee f(x_1, \ldots, x_n)) \wedge (x_0 \vee \overline{f(\bar{x}_1, \ldots, \bar{x}_n)}).\end{aligned}$$

a) If $f(x_1, \ldots, x_n)$ is any Boolean function, prove that $\hat{f}$ is self-dual.

b) Prove that $\hat{f}$ is a threshold function if and only if f is a threshold function.

103. [*HM25*] Explain how to use linear programming to test whether or not a monotone, self-dual Boolean function is a threshold function, given a list of its prime implicants. Also, if it is a threshold function, explain how to minimize the size of its representation as a majority function $\langle x_1^{w_1} \ldots x_n^{w_n}\rangle$.

104. [*25*] Apply the method of exercise 103 to find the shortest representations of the following threshold functions as majority functions: (a) $\langle x_1^2 x_2^3 x_3^5 x_4^7 x_5^{11} x_6^{13} x_7^{17} x_8^{19}\rangle$; (b) $[(x_1 x_2 x_3 x_4)_2 \geq t]$, for $0 \leq t \leq 16$ (17 cases); (c) $\langle x_1^{29} x_2^{25} x_3^{19} x_4^{15} x_5^{12} x_6^8 x_7^8 x_8^3 x_9^3 x_{10}\rangle$.

105. [*M25*] Show that the Fibonacci threshold function in exercise 101 has no shorter representation as a majority function than the one used to define it.

▶ **106.** [*M25*] The median-of-three operation $\langle x\bar{y}\bar{z}\rangle$ is true if and only if $x \geq y + z$.

a) Generalizing, show that we can test the condition $(x_1 x_2 \ldots x_n)_2 \geq (y_1 y_2 \ldots y_n)_2 + z$ by performing a median of $2^{n+1} - 1$ Boolean variables.

b) Prove that no median of fewer than $2^{n+1} - 1$ will suffice for this problem.

107. [*17*] Calculate $N(f)$ and $\Sigma(f)$ for the 16 functions in Table 1. (See Theorem T.)

108. [*M21*] Let $g(x_0, x_1, \ldots, x_n)$ be a self-dual function; thus $N(g) = 2^n$ in the notation of Theorem T. Express $N(f)$ and $\Sigma(f)$ in terms of $\Sigma(g)$, when $f(x_1, \ldots, x_n)$ is (a) $g(0, x_1, \ldots, x_n)$; (b) $g(1, x_1, \ldots, x_n)$.

109. [*M25*] The binary string $\alpha = a_1 \ldots a_n$ is said to *majorize* the binary string $\beta = b_1 \ldots b_n$, written $\alpha \succeq \beta$ or $\beta \preceq \alpha$, if $a_1 + \cdots + a_k \geq b_1 + \cdots + b_k$ for all $k \geq 0$.

a) Let $\bar{\alpha} = \bar{a}_1 \ldots \bar{a}_n$. Show that $\alpha \succeq \beta$ if and only if $\bar{\beta} \succeq \bar{\alpha}$.

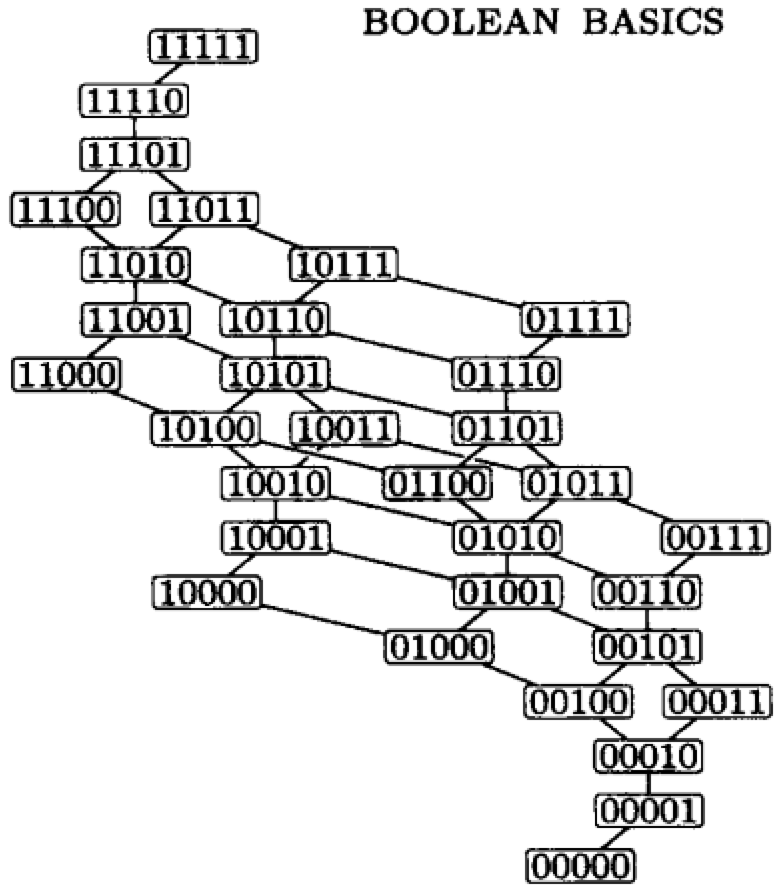


Fig. 8. The binary majorization lattice for strings of length 5. (See exercise 109.)

b) Show that any two binary strings of length n have a greatest lower bound $\alpha \wedge \beta$, which has the property that $\alpha \succeq \gamma$ and $\beta \succeq \gamma$ if and only if $\alpha \wedge \beta \succeq \gamma$. Explain how to compute $\alpha \wedge \beta$, given α and β.

c) Similarly, explain how to compute a least upper bound $\alpha \vee \beta$, with the property that $\gamma \succeq \alpha$ and $\gamma \succeq \beta$ if and only if $\gamma \succeq \alpha \vee \beta$.

d) True or false: $\alpha \wedge (\beta\vee\gamma) = (\alpha\wedge\beta) \vee (\alpha\wedge\gamma)$; $\alpha \vee (\beta\wedge\gamma) = (\alpha\vee\beta) \wedge (\alpha\vee\gamma)$.

e) Say that α *covers* β if $\alpha \succeq \beta$ and $\alpha \neq \beta$, and if $\alpha \succeq \gamma \succeq \beta$ implies that we have either $\gamma = \alpha$ or $\gamma = \beta$. For example, Fig. 8 illustrates the covering relations between binary strings of length 5. Find a simple way to describe the strings that are covered by a given binary string.

f) Show that every path $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_r = 0\ldots0$ from a given string α to $0\ldots0$, where α_{j-1} covers α_j for $1 \le j \le r$, has the same length $r = r(\alpha)$.

g) Let $m(\alpha)$ be the number of binary strings β with $\beta \succeq \alpha$. Prove that $m(1\alpha) = m(\alpha)$ and $m(0\alpha) = m(\alpha) + m(\alpha')$, where α' is α with its leftmost 1 (if any) changed to 0.

h) How many strings α of length n satisfy $\bar{\alpha} \succeq \alpha$?

110. [*M23*] A Boolean function is called *regular* if $x \preceq y$ implies that $f(x) \le f(y)$ for all vectors x and y, where $\preceq$ is the majorization relation in exercise 109. Prove or disprove the following statements:

a) Every regular function is monotone.

b) If f is a threshold function (75) for which $w_1 \ge w_2 \ge \cdots \ge w_n$, f is regular.

c) If f is as in (b) and $\Sigma(f) = (s_1, \ldots, s_n)$, then $s_1 \ge s_2 \ge \cdots \ge s_n$.

d) If f is a threshold function (75) with $s_1 \ge s_2 \ge \cdots \ge s_n$, then $w_1 \ge w_2 \ge \cdots \ge w_n$.

111. [*M36*] An *optimum coterie* for a system with working probabilities $(p_1, \ldots, p_n)$ is a coterie that corresponds to a monotone self-dual function with maximum availability, among all monotone self-dual functions with n variables. (See exercises 14 and 66.)

a) Prove that if $1 \ge p_1 \ge \cdots \ge p_n \ge \frac{1}{2}$, at least one self-dual function with maximum availability is a regular function. Describe such a function.

b) Furthermore, it suffices to test the optimality of a regular self-dual function f at points y of the binary majorization lattice for which $f(y) = 1$ but $f(x) = 0$ for all x covered by y.

c) What coterie is optimum when some probabilities are $< \frac{1}{2}$?

▶ **112.** [*M37*] (J. Håstad.) If $f(x_1, x_2, \ldots, x_m)$ is a Boolean function, let $M(f)$ be its representation as a multilinear polynomial with integer coefficients (see exercise 12). Arrange the terms in this polynomial by using Chase's sequence $\alpha_0 = 00\ldots0$, $\alpha_1 = 10\ldots0$, $\ldots$, $\alpha_{2^m-1} = 11\ldots1$ to order the exponents; Chase's sequence, obtained by concatenating the sequences A_{n0}, $A_{(n-1)1}$, $\ldots$, A_{0n} of 7.2.1.3–(35), has the nice property that α_j is identical to α_{j+1} except for a slight change, either $0 \to 1$ or $01 \to 10$ or $001 \to 100$ or $10 \to 01$ or $100 \to 001$. For example, Chase's sequence is

$$0000, 1000, 0010, 0001, 0100, 1100, 1010, 1001, 0011, 0101, 0110, 1110, 1101, 1011, 0111, 1111$$

when $m = 4$, corresponding to the respective terms 1, x_1, x_3, x_4, x_2, x_1x_2, $\ldots$, $x_2x_3x_4$, $x_1x_2x_3x_4$; so the relevant representation of, say, $((x_1 \oplus \bar{x}_2) \wedge x_3) \vee (x_1 \wedge \bar{x}_3 \wedge x_4)$ is

$$x_3 - x_1x_3 + x_1x_4 - x_2x_3 + 2x_1x_2x_3 - x_1x_3x_4$$

when the terms have been arranged in this order. Now let

$$F(f) = [\text{the most significant coefficient of } M(f) \text{ is positive}].$$

For example, the most significant (final) nonzero term of $((x_1 \oplus \bar{x}_2) \wedge x_3) \vee (x_1 \wedge \bar{x}_3 \wedge x_4)$ is $-x_1x_3x_4$ in Chase's ordering, so $F(f) = 0$ in this case.

a) Determine $F(f)$ for each of the 16 functions in Table 1.

b) Show that $F(f)$ is a threshold function of the $n = 2^m$ entries $\{f_{0\ldots00}, f_{0\ldots01}, \ldots, f_{1\ldots11}\}$ of the truth table for f. Write this function out explicitly when $m = 2$.

c) Prove that, when m is large, all the weights in any threshold representation of F must be huge: Their absolute values must all exceed

$$\frac{3^{\binom{m}{3}}\, 7^{\binom{m}{4}}\, 15^{\binom{m}{5}} \ldots (2^{m-1}-1)^{\binom{m}{m}}}{n}\bigl(1-O(n^{-1})\bigr) = 2^{mn/2-n-2(3/2)^m/\ln 2+O((5/4)^m)}.$$

Hint: Consider discrete Fourier transforms of the truth table entries.

113. [*24*] Show that the following three threshold operations suffice to evaluate the function $S_{2,3,6,8,9}(x_1, \ldots, x_{12})$ in (91):

$$g_1(x_1, \ldots, x_{12}) = [\nu x \ge 6] = \langle 1 x_1 \ldots x_{12} \rangle;$$
$$g_2(x_1, \ldots, x_{12}) = [\nu x - 6g_1 \ge 2] = \langle 1^3 x_1 \ldots x_{12} \bar{g}_1^6 \rangle;$$
$$g_3(x_1, \ldots, x_{12}) = [-2\nu x + 13g_1 + 7g_2 \ge 1] = \langle 0^5 \bar{x}_1^2 \ldots \bar{x}_{12}^2 g_1^{13} g_2^7 \rangle.$$

Also find a four-threshold scheme that evaluates $S_{1,3,5,8}(x_1, \ldots, x_{12})$.

114. [*20*] (D. A. Huffman.) What is the function $S_{3,6}(x, x, x, x, y, y, z)$?

115. [*M22*] Explain why (92) correctly computes the parity function $x_0 \oplus x_1 \oplus \cdots \oplus x_{2m}$.

▶ **116.** [*HM28*] (B. Dunham and R. Fridshal, 1957.) By considering symmetric functions, one can prove that Boolean functions of n variables might have many prime implicants.

a) Suppose $0 \le j \le k \le n$. For which symmetric functions $f(x_1, \ldots, x_n)$ is the term $x_1 \wedge \cdots \wedge x_j \wedge \bar{x}_{j+1} \wedge \cdots \wedge \bar{x}_k$ a prime implicant?

b) How many prime implicants does the function $S_{3,4,5,6}(x_1, \ldots, x_9)$ have?

c) Let $\hat{b}(n)$ be the maximum number of prime implicants, over all symmetric Boolean functions of n variables. Find a recurrence formula for $\hat{b}(n)$, and compute $\hat{b}(9)$.

d) Prove that $\hat{b}(n) = \Theta(3^n/n)$.

e) Show that, furthermore, there are symmetric functions $f(x_1, \ldots, x_n)$ for which both f and $\bar{f}$ have $\Theta(2^{3n/2}/n)$ prime implicants.

117. [*M26*] A disjunctive normal form is called *irredundant* if none of its implicants implies another. Let $b^*(n)$ be the maximum number of implicants in an irredundant

DNF, over all Boolean functions of n variables. Find a simple formula for $b^*(n)$, and determine its asymptotic value.

118. [*29*] How many Boolean functions $f(x_1, x_2, x_3, x_4)$ have exactly m prime implicants, for $m = 0, 1, \ldots$?

119. [*M48*] Continuing the previous exercises, let $b(n)$ be the maximum number of prime implicants in a Boolean function of n variables. Clearly $\hat{b}(n) \le b(n) < b^*(n)$; what is the asymptotic value of $b(n)$?

120. [*23*] What is the shortest DNF for the symmetric functions (a) $x_1 \oplus x_2 \oplus \cdots \oplus x_n$? (b) $S_{0,1,3,4,6,7}(x_1, \ldots, x_7)$? (c) Prove that every Boolean function of n variables can be expressed as a DNF with at most 2^{n-1} prime implicants.

▶ **121.** [*M23*] The function $\langle 1 (x_1 \oplus x_2) y_1 y_2 y_3 \rangle$ is partially symmetric, since it is symmetric in $\{x_1, x_2\}$ and in $\{y_1, y_2, y_3\}$, but not in all five variables $\{x_1, x_2, y_1, y_2, y_3\}$.

a) Exactly how many Boolean functions $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$ are symmetric in $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$?
b) How many of those functions are monotone?
c) How many of those functions are self-dual?
d) How many of those functions are monotone and self-dual?

122. [*M25*] Continuing exercises 110 and 121, find all Boolean functions $f(x_1, x_2, x_3, y_1, y_2, y_3, y_4, y_5, y_6)$ that are simultaneously symmetric in $\{x_1, x_2, x_3\}$, symmetric in $\{y_1, y_2, \ldots, y_6\}$, self-dual, and regular. Which of them are threshold functions?

123. [*46*] Determine the exact number of self-dual Boolean functions of ten variables that are threshold functions.

124. [*20*] Find a Boolean function of four variables that is equivalent to 767 other functions, under the ground rules of Table 5.

125. [*18*] Which of the function classes in (95) are canalizing?

126. [*23*] (a) Show that a Boolean function is canalizing if and only if its sets of prime implicants and prime clauses have a certain simple property. (b) Show that a Boolean function is canalizing if and only if its Chow parameters $N(f)$ and $\Sigma(f)$ have a certain simple property (see Theorem T). (c) Define the Boolean vectors

$$\vee(f) = \bigvee\{x \mid f(x) = 1\} \qquad \text{and} \qquad \wedge(f) = \bigwedge\{x \mid f(x) = 1\};$$

by analogy with the integer vector $\Sigma(f)$. Show that it's possible to decide whether or not f is canalizing, given only the four vectors $\vee(f)$, $\vee(\bar{f})$, $\wedge(f)$, and $\wedge(\bar{f})$.

127. [*M25*] Which canalizing functions are (a) self-dual? (b) definite Horn functions?

▶ **128.** [*20*] Find a noncanalizing $f(x_1, \ldots, x_n)$ that is true at exactly two points.

129. [*M25*] How many different canalizing functions of n variables exist?

130. [*M21*] According to Table 3, there are 168 monotone Boolean functions of four variables. But some of them, like $x \wedge y$, depend on only three variables or fewer.

a) How many 4-variable monotone Boolean functions actually involve each variable?
b) How many of those functions are distinct under permutation, as in Table 4?

131. [*HM42*] Table 3 makes it clear that there are many more Horn functions than Krom functions. What is the asymptotic number, as $n \to \infty$?

▶ **132.** [*HM30*] The Boolean function $g(x) = g(x_1, \ldots, x_n)$ is called *affine* if it can be written in the form $y_0 \oplus (x_1 \wedge y_1) \oplus \cdots \oplus (x_n \wedge y_n) = (y_0 + x \cdot y) \bmod 2$ for some Boolean constants $y_0, y_1, \ldots, y_n$.

a) Given any Boolean function $f(x)$, show that some affine function agrees with $f(x)$ at $2^{n-1} + 2^{n/2-1}$ or more points x. *Hint:* Let $s(y) = \sum_x (-1)^{f(x)+x\cdot y}$, and prove that $\sum_y s(y)s(y \oplus z) = 2^{2n}[z=0\ldots0]$ for all vectors z.

b) The Boolean function $f(x)$ is called *bent* if no affine function agrees with it at more than $2^{n-1} + 2^{n/2-1}$ points. Prove that

$$(x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus \cdots \oplus (x_{n-1} \wedge x_n) \oplus h(x_2, x_4, \ldots, x_n)$$

is a bent function, when n is even and $h(y_1, y_2, \ldots, y_{n/2})$ is arbitrary.

c) Prove that $f(x)$ is a bent function if and only if

$$\sum_x (f(x) \oplus f(x \oplus y)) = 2^{n-1} \qquad \text{for all } y \neq 0\ldots0.$$

d) If a bent function $f(x_1, \ldots, x_n)$ is represented by a multilinear polynomial mod 2 as in (19), show that it never contains the term $x_1 \ldots x_r$ when $r > n/2 > 1$.

▸ **133.** [*20*] (Mark A. Smith, 1990.) Suppose we flip n independent coins to get n random bits, where the kth coin produces bit 1 with probability p_k. Find a way to choose $(p_1, \ldots, p_n)$ so that $f(x_1, \ldots, x_n) = 1$ with probability $(t_0 t_1 \ldots t_{2^n-1})_2/(2^{2^n}-1)$, where $t_0 t_1 \ldots t_{2^n-1}$ is the truth table of the Boolean function f. (Thus, n suitable random coins can generate a probability with 2^n-bit precision.)

By and large the minimization of switching components outweighs all other engineering considerations in designing economical logic circuits.

— H. A. CURTIS, *A New Approach to the Design of Switching Circuits* (1962)

He must be a great calculator indeed who succeeds. Simplify, simplify.

— HENRY D. THOREAU, *Walden; or, Life in the Woods* (1854)

7.1.2. Boolean Evaluation

Our next goal is to study the efficient evaluation of Boolean functions, much as we studied the evaluation of polynomials in Section 4.6.4. One natural way to investigate this topic is to consider chains of basic operations, analogous to the polynomial chains discussed in that section.

A *Boolean chain*, for functions of n variables $(x_1, \ldots, x_n)$, is a sequence $(x_{n+1}, \ldots, x_{n+r})$ with the property that each step combines two of the preceding steps:

$$x_i \;=\; x_{j(i)} \circ_i x_{k(i)}, \qquad \text{for } n+1 \le i \le n+r, \tag{1}$$

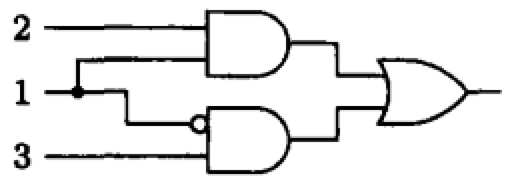
where $1 \le j(i) < i$ and $1 \le k(i) < i$, and where $\circ_i$ is one of the sixteen binary operators of Table 7.1.1–1. For example, when $n = 3$ the two chains

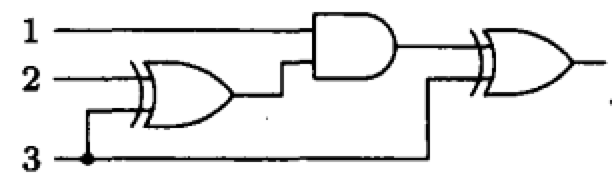
$$\begin{aligned} x_4 &= x_1 \wedge x_2 \\ x_5 &= \bar{x}_1 \wedge x_3 \\ x_6 &= x_4 \vee x_5 \end{aligned} \qquad \text{and} \qquad \begin{aligned} x_4 &= x_2 \oplus x_3 \\ x_5 &= x_1 \wedge x_4 \\ x_6 &= x_3 \oplus x_5 \end{aligned} \tag{2}$$

both evaluate the "mux" or "if-then-else" function $x_6 = (x_1?\ x_2\colon x_3)$, which takes the value x_2 or x_3 depending on whether x_1 is 1 (true) or 0 (false).

(Notice that the left-hand example in (2) uses the simplified notation '$x_5 = \bar{x}_1 \wedge x_3$' to specify the NOT-BUT operation, instead of the form '$x_5 = x_1 \overline{\subset} x_3$' that appears in Table 7.1.1–1. The main point is that, regardless of notation, every step of a Boolean chain is a Boolean combination of two prior results.)

Boolean chains correspond naturally to electronic circuits, with each step in the chain corresponding to a "gate" that has two inputs and one output. Electrical engineers traditionally represent the Boolean chains of (2) by circuit diagrams such as

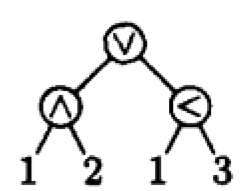


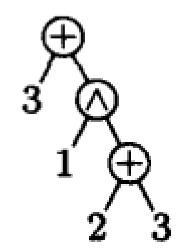
and


. (3)

They need to design economical circuits that are subject to various technological constraints; for example, some gates might be more expensive than others, some outputs might need to be amplified if reused, the layout might need to be planar or nearly so, some paths might need to be short. But our chief concern in this book is software, not hardware, so we don't have to worry about such things. For our purposes, all gates have equal cost, and all outputs can be reused as often as desired. (Jargonwise, our Boolean chains boil down to circuits in which all gates have fan-in 2 and unlimited fan-out.)

Furthermore we shall depict Boolean chains as binary trees such as



and


(4)

instead of using circuit diagrams like (3). Such binary trees will have overlapping subtrees when intermediate steps of the chain are used more than once. Every internal node is labeled with a binary operator; external nodes are labeled with an integer k, representing the variable x_k. The label '$\circledless$' in the left tree of (4) stands for the NOT-BUT operator, since $\bar{x} \wedge y = [x < y]$; similarly, the BUT-NOT operator, $x \wedge \bar{y}$, can be represented by the node label '$\circledgtr$'.

Several different Boolean chains might have the same tree diagram. For example, the left-hand tree of (4) also represents the chain

$$x_4 = \bar{x}_1 \wedge x_3, \qquad x_5 = x_1 \wedge x_2, \qquad x_6 = x_5 \vee x_4.$$

Any topological sorting of the tree nodes yields an equivalent chain.

Given a Boolean function f of n variables, we often want to find a Boolean chain such that $x_{n+r} = f(x_1, \ldots, x_n)$, where r is as small as possible. The *combinational complexity* $C(f)$ of a function f is the length of the shortest chain that computes it. To save excess verbiage, we will simply call $C(f)$ the "cost of f." The mux function in our examples above has cost 3, because one can show by exhaustive trials that it can't be produced by any Boolean chain of length 2.

The DNF and CNF representations of f, which we studied in Section 7.1.1, rarely tell us much about $C(f)$, since substantially more efficient schemes of

calculation are usually possible. For example, in the discussion following 7.1.1–(30) we found that the more-or-less random function of four variables whose truth table is 1100 1001 0000 1111 has no DNF expression shorter than

$$(\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge \bar{x}_4) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2). \tag{5}$$

This formula corresponds to a Boolean chain of 10 steps. But that function can also be expressed more cleverly as

$$\big(((x_2 \wedge \bar{x}_4) \oplus \bar{x}_3) \wedge \bar{x}_1\big) \oplus x_2, \tag{6}$$

so its complexity is at most 4.

How can nonobvious formulas like (6) be discovered? We will see that a computer can find the best chains for functions of four variables without doing an enormous amount of work. Still, the results can be quite startling, even for people who have had considerable experience with Boolean algebra. Typical examples of this phenomenon can be seen in Fig. 9, which illustrates the four-variable functions that are perhaps of greatest general interest, namely the functions that are symmetric under all permutations of their variables.

Consider, for example, the function $S_2(x_1, x_2, x_3, x_4)$, for which we have

x_1	0000 0000 1111 1111	
x_2	0000 1111 0000 1111	
x_3	0011 0011 0011 0011	
x_4	0101 0101 0101 0101	
$x_5 = x_1 \oplus x_3$	0011 0011 1100 1100	
$x_6 = x_1 \oplus x_2$	0000 1111 1111 0000	(7)
$x_7 = x_3 \oplus x_4$	0110 0110 0110 0110	
$x_8 = x_5 \vee x_6$	0011 1111 1111 1100	
$x_9 = x_6 \oplus x_7$	0110 1001 1001 0110	
$x_{10} = x_8 \wedge \bar{x}_9$	0001 0110 0110 1000	

according to Fig. 9. Truth tables are shown here so that we can easily verify each step of the calculation. Step x_8 yields a function that is true whenever $x_1 \neq x_2$ or $x_1 \neq x_3$; and $x_9 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ is the parity function $(x_1 + x_2 + x_3 + x_4) \bmod 2$. Therefore the final result, x_{10}, is true precisely when exactly two of $\{x_1, x_2, x_3, x_4\}$ are 1; these are the cases that satisfy x_8 and have even parity.

Several of the other computational schemes of Fig. 9 can also be justified intuitively. But some of the chains, like the one for $S_{1,4}$, are quite amazing.

Notice that the intermediate result x_6 is used twice in (7). In fact, no six-step chain for the function $S_2(x_1, x_2, x_3, x_4)$ is possible without making double use of some intermediate subexpression; the shortest algebraic formulas for S_2, including nice symmetrical ones like

$$\big((x_1 \wedge x_2) \vee (x_3 \wedge x_4)\big) \oplus \big((x_1 \vee x_2) \wedge (x_3 \vee x_4)\big), \tag{8}$$

all have cost 7. But Fig. 9 shows that the other symmetric functions of four variables can all be evaluated optimally via "pure" binary trees, without overlapping subtrees except at external nodes (which represent the variables).

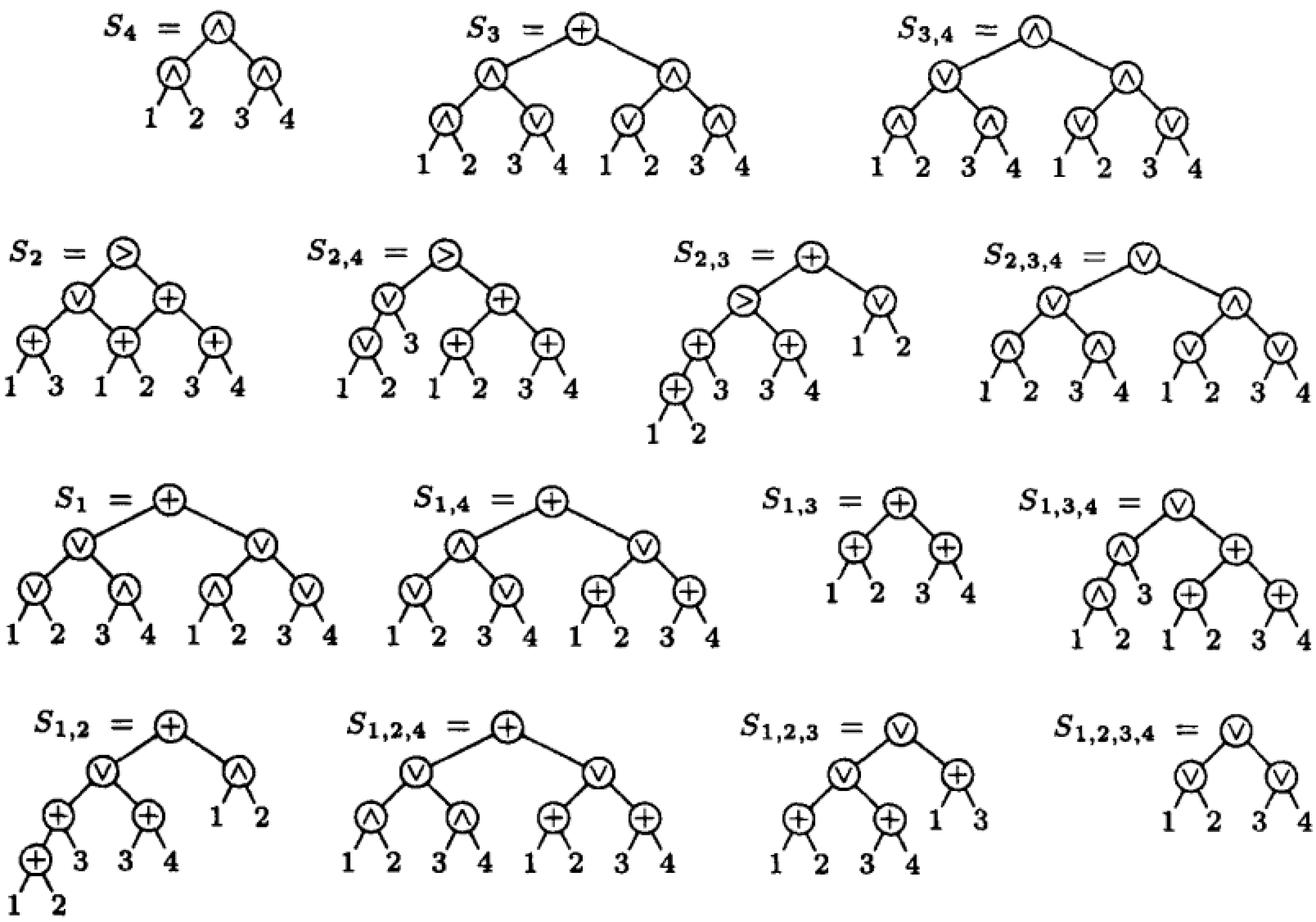


Fig. 9. Optimum Boolean chains for the symmetric functions of four variables.

In general, if $f(x_1, \ldots, x_n)$ is any Boolean function, we say that its *length* $L(f)$ is the number of binary operators in the shortest formula for f. Obviously $L(f) \geq C(f)$; and we can easily verify that $L(f) = C(f)$ whenever $n \leq 3$, by considering the fourteen basic types of 3-variable functions in 7.1.1–(95). But we have just seen that $L(S_2) = 7$ exceeds $C(S_2) = 6$ when $n = 4$, and in fact $L(f)$ is almost always substantially larger than $C(f)$ when n is large (see exercise 49).

The *depth* $D(f)$ of a Boolean function f is another important measure of its inherent complexity: We say that the depth of a Boolean chain is the length of the longest downward path in its tree diagram, and $D(f)$ is the minimum achievable depth when all Boolean chains for f are considered. All of the chains illustrated in Fig. 9 have not only the minimum cost but also the minimum depth — except in the cases $S_{2,3}$ and $S_{1,2}$, where we cannot simultaneously achieve cost 6 and depth 3. The formula

$$S_{2,3}(x_1, x_2, x_3, x_4) = \big((x_1 \wedge x_2) \oplus (x_3 \wedge x_4)\big) \vee \big((x_1 \vee x_2) \wedge (x_3 \oplus x_4)\big) \qquad (9)$$

shows that $D(S_{2,3}) = 3$, and a similar formula works for $S_{1,2}$.

Optimum chains for $n = 4$. Exhaustive computations for 4-variable functions are feasible because such functions have only $2^{16} = 65{,}536$ possible truth tables. In fact we need only consider half of those truth tables, because the complement $\bar{f}$ of any function f has the same cost, length, and depth as f itself.

Let's say that $f(x_1,\ldots,x_n)$ is *normal* if $f(0,\ldots,0)=0$, and in general that

$$f(x_1,\ldots,x_n) \oplus f(0,\ldots,0) \tag{10}$$

is the "normalization" of f. Any Boolean chain can be normalized by normalizing each of its steps and by making appropriate changes to the operators; for if $(\hat{x}_1,\ldots,\hat{x}_{i-1})$ are the normalizations of $(x_1,\ldots,x_{i-1})$ and if $x_i = x_{j(i)} \circ_i x_{k(i)}$ as in (1), then $\hat{x}_i$ is clearly a binary function of $\hat{x}_{j(i)}$ and $\hat{x}_{k(i)}$. (Exercise 7 presents an example.) Therefore we can restrict consideration to normal Boolean chains, without loss of generality.

Notice that a Boolean chain is normal if and only if each of its binary operators $\circ_i$ is normal. And there are only eight normal binary operators — three of which, namely $\bot$, L, and R, are trivial. So we can assume that all Boolean chains of interest are formed from the five operators $\wedge$, $\overline{\subset}$, $\overline{\supset}$, $\vee$, and $\oplus$, which are denoted respectively by ⓐ, ⓒ, ⓓ, ⓥ, and ⊕ in Fig. 9. Furthermore we can assume that $j(i) < k(i)$ in each step.

There are $2^{15} = 32{,}768$ normal functions of four variables, and we can compute their lengths without difficulty by systematically enumerating all functions of length 0, 1, 2, etc. Indeed, $L(f)=r$ implies that $f = g \circ h$ for some g and h, where $L(g)+L(h) = r-1$ and $\circ$ is one of the five nontrivial normal operators; so we can proceed as follows:

Algorithm L (*Find normal lengths*). This algorithm determines $L(f)$ for all normal truth tables $0 \le f < 2^{2^n-1}$, by building lists of all nonzero normal functions of length r for $r \ge 0$.

L1. [Initialize.] Let $L(0) \leftarrow 0$ and $L(f) \leftarrow \infty$ for $1 \le f < 2^{2^n-1}$. Then, for $1 \le k \le n$, set $L(x_k) \leftarrow 0$ and put x_k into list 0, where

$$x_k = (2^{2^n}-1)/(2^{2^{n-k}}+1) \tag{11}$$

is the truth table for x_k. (See exercise 8.) Finally, set $c \leftarrow 2^{2^n-1}-n-1$; c is the number of places where $L(f)=\infty$.

L2. [Loop on r.] Do step L3 for $r = 1, 2, \ldots$; eventually the algorithm will terminate when c becomes 0.

L3. [Loop on j and k.] Do step L4 for $j = 0, 1, \ldots$, and $k = r-1-j$, while $j \le k$.

L4. [Loop on g and h.] Do step L5 for all g in list j and all h in list k. (If $j=k$, it suffices to restrict h to functions that *follow* g in list k.)

L5. [Loop on f.] Do step L6 for $f = g \mathbin{\&} h$, $f = \bar{g} \mathbin{\&} h$, $f = g \mathbin{\&} \bar{h}$, $f = g \mid h$, and $f = g \oplus h$. (Here $g \mathbin{\&} h$ denotes the bitwise AND of the integers g and h; we are representing truth tables by integers in binary notation.)

L6. [Is f new?] If $L(f) = \infty$, set $L(f) \leftarrow r$, $c \leftarrow c-1$, and put f in list r. Terminate the algorithm if $c=0$. ▌

Exercise 10 shows that a similar procedure will compute all depths $D(f)$.

With a little more work, we can in fact modify Algorithm L so that it finds better upper bounds on $C(f)$, by computing a heuristic bit vector $\phi(f)$ called

Table 1

THE NUMBER OF FOUR-VARIABLE FUNCTIONS WITH GIVEN COMPLEXITY

$C(f)$	Classes	Functions	$L(f)$	Classes	Functions	$D(f)$	Classes	Functions
0	2	10	0	2	10	0	2	10
1	2	60	1	2	60	1	2	60
2	5	456	2	5	456	2	17	1458
3	20	2474	3	20	2474	3	179	56456
4	34	10624	4	34	10624	4	22	7552
5	75	24184	5	75	24184	5	0	0
6	72	25008	6	68	24640	6	0	0
7	12	2720	7	16	3088	7	0	0

the "footprint" of f. A normal Boolean chain can begin in only $5\binom{n}{2}$ different ways, since the first step x_{n+1} must be either $x_1 \wedge x_2$ or $\bar{x}_1 \wedge x_2$ or $x_1 \wedge \bar{x}_2$ or $x_1 \vee x_2$ or $x_1 \oplus x_2$ or $x_1 \wedge x_3$ or $\cdots$ or $x_{n-1} \oplus x_n$. Suppose $\phi(f)$ is a bit vector of length $5\binom{n}{2}$ and $U(f)$ is an upper bound on $C(f)$, with the following property: Every 1 bit in $\phi(f)$ corresponds to the first step of some Boolean chain that computes f in $U(f)$ steps.

Such pairs $(U(f), \phi(f))$ can be computed by extending the basic strategy of Algorithm L. Initially we set $U(f) \leftarrow 1$ and we set $\phi(f)$ to an appropriate vector $0\ldots010\ldots0$, for all functions f of cost 1. Then, for $r = 2, 3, \ldots$, we proceed to look for functions $f = g \circ h$ where $U(g) + U(h) = r - 1$, as before, but with two changes: (1) If the footprints of g and h have at least one element in common, namely if $\phi(g) \mathbin{\&} \phi(h) \neq 0$, then we know that $C(f) \leq r - 1$, so we can decrease $U(f)$ if it was $\geq r$. (2) If the cost of $g \circ h$ is equal to (but not less than) our current upper bound $U(f)$, we can set $\phi(f) \leftarrow \phi(f) \mid (\phi(g) \mid \phi(h))$ if $U(f) = r$, $\phi(f) \leftarrow \phi(f) \mid (\phi(g) \mathbin{\&} \phi(h))$ if $U(f) = r - 1$. Exercise 11 works out the details.

It turns out that this footprint heuristic is powerful enough to find chains of optimum cost $U(f) = C(f)$ for all functions f, when $n = 4$. Moreover, we'll see later that footprints also help us solve more complicated evaluation problems.

According to Table 7.1.1–5, the $2^{16} = 65{,}536$ functions of four variables belong to only 222 distinct classes when we ignore minor differences due to permutation of variables and/or complementation of values. Algorithm L and its variants lead to the overall statistics shown in Table 1.

***Evaluation with minimum memory.** Suppose the Boolean values $x_1, \ldots, x_n$ appear in n registers, and we want to evaluate a function by performing a sequence of operations having the form

$$x_{j(i)} \leftarrow x_{j(i)} \circ_i x_{k(i)}, \qquad \text{for } 1 \leq i \leq r, \tag{12}$$

where $1 \leq j(i) \leq n$ and $1 \leq k(i) \leq n$ and $\circ_i$ is a binary operator. At the end of the computation, the desired function value should appear in one of the registers. When $n = 3$, for example, the four-step sequence

$$\begin{array}{ll} & (x_1 = 00001111 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_1 \leftarrow x_1 \oplus x_2 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_3 \leftarrow x_3 \wedge x_1 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 00010100) \\ x_2 \leftarrow x_2 \wedge \bar{x}_1 & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010100) \\ x_3 \leftarrow x_3 \vee x_2 & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010111) \end{array} \tag{13}$$

computes the median $\langle x_1x_2x_3\rangle$ and puts it into the original position of x_3. (All eight possibilities for the register contents are shown here as truth tables, before and after each operation.)

In fact we can check the calculation by working with only one truth table at a time, instead of keeping track of all three, if we analyze the situation backwards. Let $f_l(x_1,\ldots,x_n)$ denote the function computed by steps l, $l+1$, ..., r of the sequence, omitting the first $l-1$ steps; thus, in our example, $f_2(x_1,x_2,x_3)$ would be the result in x_3 after the three steps $x_3 \leftarrow x_3\wedge x_1$, $x_2 \leftarrow x_2\wedge\bar{x}_1$, $x_3 \leftarrow x_3\vee x_2$. Then the function computed in register x_3 by all four steps is

$$f_1(x_1,x_2,x_3) \;=\; f_2(x_1\oplus x_2,x_2,x_3). \tag{14}$$

Similarly $f_2(x_1,x_2,x_3) = f_3(x_1,x_2,x_3\wedge x_1)$, $f_3(x_1,x_2,x_3) = f_4(x_1,x_2\wedge\bar{x}_1,x_3)$, $f_4(x_1,x_2,x_3) = f_5(x_1,x_2,x_3\vee x_2)$, and $f_5(x_1,x_2,x_3) = x_3$. We can therefore go back from f_5 to f_4 to $\cdots$ to f_1 by operating on truth tables in an appropriate way.

For example, suppose $f(x_1,x_2,x_3)$ is a function whose truth table is

$$t \;=\; a_0a_1a_2a_3a_4a_5a_6a_7;$$

then the truth table for $g(x_1,x_2,x_3) = f(x_1\oplus x_2,x_2,x_3)$ is

$$u \;=\; a_0a_1a_6a_7a_4a_5a_2a_3,$$

obtained by replacing a_x by $a_{x'}$, where

$$x=(x_1x_2x_3)_2 \qquad\text{implies}\qquad x'=((x_1\oplus x_2)x_2x_3)_2.$$

Similarly the truth table for, say, $h(x_1,x_2,x_3) = f(x_1,x_2,x_3\wedge x_1)$ is

$$v \;=\; a_0a_0a_2a_2a_4a_5a_6a_7.$$

And we can use bitwise operations to compute u and v from t:

$$u = t\oplus\bigl((t\oplus(t\gg 4)\oplus(t\ll 4))\,\&\,(00110011)_2\bigr), \tag{15}$$
$$v = t\oplus\bigl((t\oplus(t\gg 1))\,\&\,(01010000)_2\bigr). \tag{16}$$

Let $C_m(f)$ be the length of a shortest minimum-memory computation for f. The backward-computation principle tells us that, if we know the truth tables of all functions f with $C_m(f) < r$, we can readily find all the truth tables of functions with $C_m(f) = r$. Namely, we can restrict consideration to normal functions as before. Then, for all normal g such that $C_m(g) = r-1$, we can construct the $5n(n-1)$ truth tables for

$$g(x_1,\ldots,x_{j-1},x_j\circ x_k,x_{j+1},\ldots,x_n) \tag{17}$$

and mark them with cost r if they haven't previously been marked. Exercise 14 shows that those truth tables can all be computed by performing simple bitwise operations on the truth table for g.

When $n = 4$, all but 13 of the 222 basic function types turn out to have $C_m(f) = C(f)$, so they can be evaluated in minimum memory without increasing the cost. In particular, all of the symmetric functions have this property—although that fact is not at all obvious from Fig. 9. Five classes of functions

have $C(f) = 5$ but $C_m(f) = 6$; eight classes have $C(f) = 6$ but $C_m(f) = 7$. The most interesting example of the latter type is probably the function $(x_1 \vee x_2) \oplus (x_3 \vee x_4) \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$, which has cost 6 because of the formula

$$x_1 \oplus (x_3 \vee x_4) \oplus \bigl(x_2 \wedge (\bar{x}_1 \vee (x_3 \wedge x_4))\bigr), \qquad (18)$$

but it has no minimum-memory chain of length less than 7. (See exercise 15.)

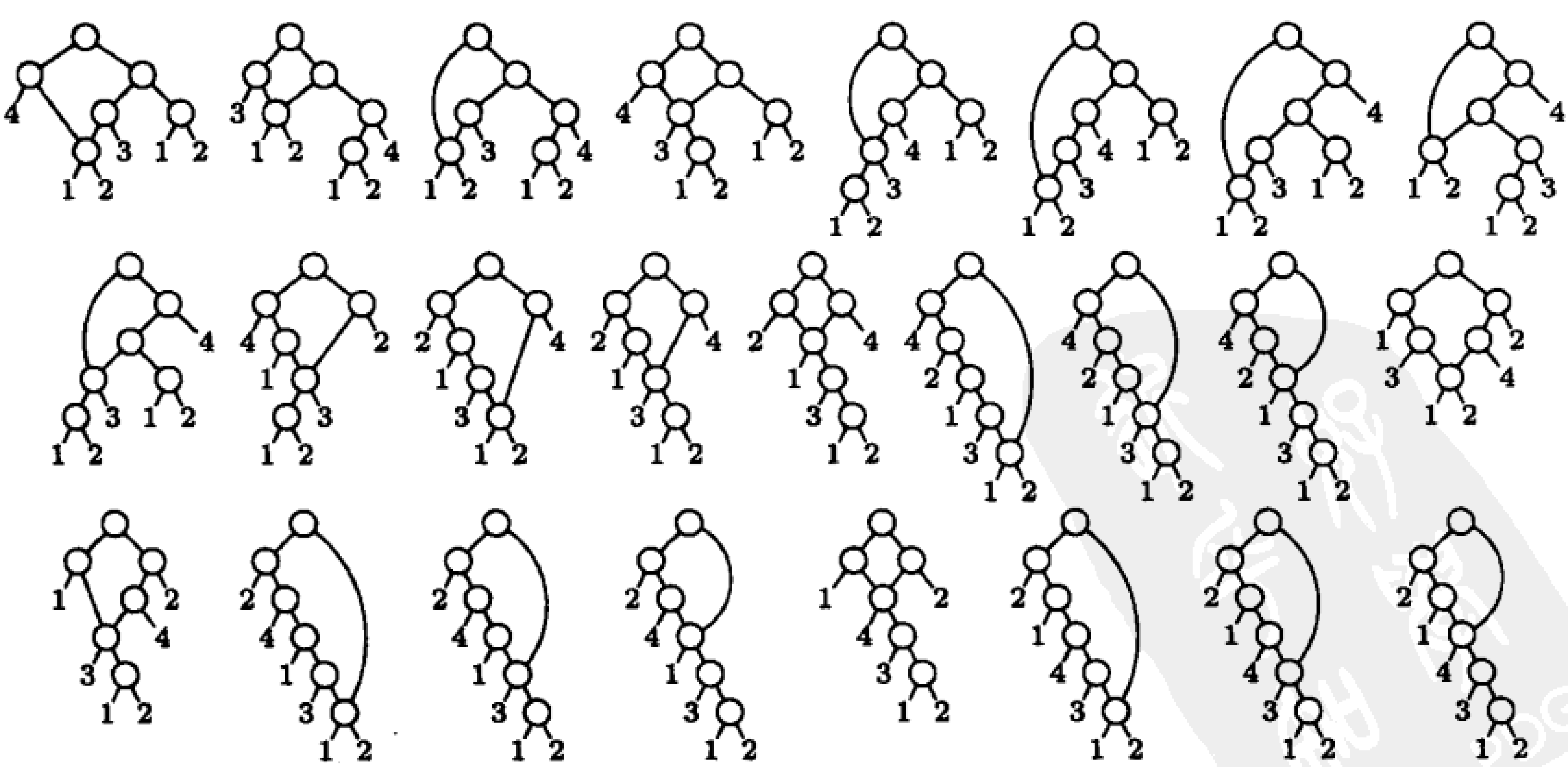
***Determining the minimum cost.** The exact value of $C(f)$ can be found by observing that all optimum Boolean chains $(x_{n+1}, \ldots, x_{n+r})$ for f obviously satisfy at least one of three conditions:

i) $x_{n+r} = x_j \circ x_k$, where x_j and x_k use no common intermediate results;
ii) $x_{n+1} = x_j \circ x_k$, where either x_j or x_k is not used in steps $x_{n+2}, \ldots, x_{n+r}$;
iii) Neither of the above, even when the intermediate steps are renumbered.

In case (i) we have $f = g \circ h$, where $C(g) + C(h) = r - 1$, and we can call this a "top-down" construction. In case (ii) we have $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{j-1}, x_j \circ x_k, x_{j+1}, \ldots, x_n)$, where $C(g) = r-1$; we call this construction "bottom-up."

The best chains that recursively use only top-down constructions correspond to minimum formula length, $L(f)$. The best chains that recursively use only bottom-up constructions correspond to minimum-memory calculations, of length $C_m(f)$. We can do better yet, by mixing top-down constructions with bottom-up constructions; but we still won't know that we've found $C(f)$, because a special chain belonging to case (iii) might be shorter.

Fortunately such special chains are rare, because they must satisfy rather strong conditions, and they can be exhaustively listed when n and r aren't too large. For example, exercise 19 proves that no special chains exist when $r < n+2$; and when $n = 4$, $r = 6$, there are only 25 essentially different special chains that cannot obviously be shortened:



By systematically trying 5^r possibilities in every special chain, one for each way to assign a normal operator to the internal nodes of the tree, we will find at least

one function f in every equivalence class for which the minimum cost $C(f)$ is achievable only in case (iii).

In fact, when $n = 4$ and $r = 6$, these $25 \cdot 5^6 = 390{,}625$ trials yield only one class of functions that can't be computed in 6 steps by any top-down-plus-bottom-up chain. The missing class, typified by the partially symmetric function $(\langle x_1x_2x_3\rangle \vee x_4) \oplus (x_1\wedge x_2\wedge x_3)$, can be reached in six steps by appropriately specializing any of the first five chains illustrated above; for example, one way is

$$x_5 = x_1 \wedge x_2, \quad x_6 = x_1 \vee x_2, \quad x_7 = x_3 \oplus x_5,$$
$$x_8 = x_4 \wedge \bar{x}_5, \quad x_9 = x_6 \wedge x_7, \quad x_{10} = x_8 \vee x_9, \qquad (19)$$

corresponding to the first special chain. Since all other functions have $L(f) \le 7$, these trial calculations have established the true minimum cost in all cases.

Historical notes: The first concerted attempts to evaluate all Boolean functions $f(w,x,y,z)$ optimally were reported in *Annals of the Computation Laboratory of Harvard University* **27** (1951), where Howard Aiken's staff presented heuristic methods and extensive tables of the best switching circuits they were able to construct. Their cost measure $V(f)$ was different from the cost $C(f)$ that we've been considering, because it was based on "control grids" of vacuum tubes: They had three kinds of gates, NOR, OR, and NAND, each of which could take k inputs with cost k. Every input to such a gate could be either a variable, or the complement of a variable, or the result of a previous gate. Furthermore the function being evaluated was represented at the top level as an AND of any number of gates, with no additional cost.

With those cost criteria, a function might not have the same cost as its complement, because AND gates were possible only at the top level. One could evaluate $x\wedge y$ as $\mathrm{NOR}(\bar{x},\bar{y})$, with cost 2; but the cost of $\bar{x}\vee(\bar{y}\wedge\bar{z})=\mathrm{NAND}(x,\mathrm{OR}(y,z))$ was 4 while its complement $x \wedge (y \vee z) = \mathrm{AND}\bigl(\mathrm{NOR}(\bar{x}),\mathrm{OR}(y,z)\bigr)$ cost only 3. Therefore the Harvard researchers needed to consider 402 essentially different classes of 4-variable functions instead of 222 (see the answer to exercise 7.1.1–125). Of course in those days they were working by hand. They found $V(f) < 20$ in all cases, except for the 64 functions equivalent to $S_{0,1}(w,x,y,z) \vee \bigl(S_2(w,x,y) \wedge z\bigr)$, which they evaluated with 20 control grids as follows:

$$g_1 = \mathrm{NOR}(\bar{w},\bar{x}),\ g_2 = \mathrm{NAND}(\bar{y},z),\ g_3 = \mathrm{NOR}(w,x),$$
$$f = \mathrm{AND}\bigl(\mathrm{NAND}(g_1,g_2),\mathrm{NAND}(g_3,\mathrm{NOR}(\bar{y},\bar{z})),$$
$$\mathrm{NOR}(\mathrm{NOR}(g_3,\bar{y},z),\mathrm{NOR}(g_1,g_2,g_3))\bigr). \qquad (20)$$

The first computer program to find provably optimum circuits was written by Leo Hellerman [*IEEE Transactions* **EC-12** (1963), 198–223], who determined the fewest NOR gates needed to evaluate any given function $f(x,y,z)$. He required every input of every gate to be either an uncomplemented variable or the output of a previous gate; fan-in and fan-out were limited to at most 3. When two circuits had the same gate count, he preferred the one with smallest sum-of-inputs. For example, he computed $\bar{x} = \mathrm{NOR}(x)$ with cost 1; $x \vee y \vee z = \mathrm{NOR}(\mathrm{NOR}(x,y,z))$ with cost 2; $\langle xyz\rangle = \mathrm{NOR}(\mathrm{NOR}(x,y),\mathrm{NOR}(x,z),\mathrm{NOR}(y,z))$

Table 2

THE NUMBER OF FIVE-VARIABLE FUNCTIONS WITH GIVEN COMPLEXITY

$C(f)$	Classes	Functions	$L(f)$	Classes	Functions	$D(f)$	Classes	Functions
0	2	12	0	2	12	0	2	12
1	2	100	1	2	100	1	2	100
2	5	1140	2	5	1140	2	17	5350
3	20	11570	3	20	11570	3	1789	6702242
4	93	109826	4	93	109826	4	614316	4288259592
5	389	995240	5	366	936440	5	0	0
6	1988	8430800	6	1730	7236880	6	0	0
7	11382	63401728	7	8782	47739088	7	0	0
8	60713	383877392	8	40297	250674320	8	0	0
9	221541	1519125536	9	141422	955812256	9	0	0
10	293455	2123645248	10	273277	1945383936	10	0	0
11	26535	195366784	11	145707	1055912608	11	0	0
12	1	1920	12	4423	31149120	12	0	0

with cost 4; $S_1(x,y,z) = \mathrm{NOR}\big(\mathrm{NOR}(x,y,z), \langle xyz\rangle\big)$ with cost 6; etc. Since he limited the fan-out to 3, he found that every function of three variables could be evaluated with cost 7 or less, except for the parity function $x\oplus y\oplus z = (x\equiv y)\equiv z$, where $x\equiv y$ has cost 4 because it is $\mathrm{NOR}(\mathrm{NOR}(x,\mathrm{NOR}(x,y)),\mathrm{NOR}(y,\mathrm{NOR}(x,y)))$.

Electrical engineers continued to explore other cost criteria; but four-variable functions seemed out of reach until 1977, when Frank M. Liang established the values of $C(f)$ shown in Table 1. Liang's unpublished derivation was based on a study of all chains that cannot be reduced by the bottom-up construction.

The case $n = 5$. There are 616,126 classes of essentially different functions $f(x_1,x_2,x_3,x_4,x_5)$, according to Table 7.1.1–5. Computers are now fast enough that this number is no longer frightening; so the author decided while writing this section to investigate $C(f)$ for all Boolean functions of five variables. Thanks to a bit of good luck, complete results could indeed be obtained, leading to the statistics shown in Table 2.

For this calculation Algorithm L and its variants were modified to deal with class representatives, instead of with the entire set of 2^{31} normal truth tables. The method of exercise 7.2.1.2–20 made it easy to generate all functions of a class, given any one of them, resulting in a thousand-fold speedup. The bottom-up method was enhanced slightly, allowing it to deduce for example that $f(x_1\wedge x_2, x_1\vee x_2, x_3, x_4, x_5)$ has cost $\le r$ if $C(f) = r-2$. After all classes of cost 10 had been found, the top-down and bottom-up methods were able to find chains of length ≤ 11 for all but seven classes of functions. Then the time-consuming part of the computation began, in which approximately 53 million special chains with $n = 5$ and $r = 11$ were generated; every such chain led to $5^{11} = 48{,}828{,}125$ functions, some of which would hopefully fall into the seven remaining mystery classes. But only six of those classes were found to have 11-step solutions. The lone survivor, whose truth table is `169ae443` in hexadecimal notation, is the unique class for which $C(f) = 12$, and it also has $L(f) = 12$.

The resulting constructions of symmetric functions are shown in Fig. 10. Some of them are astonishingly beautiful; some of them are beautifully simple;

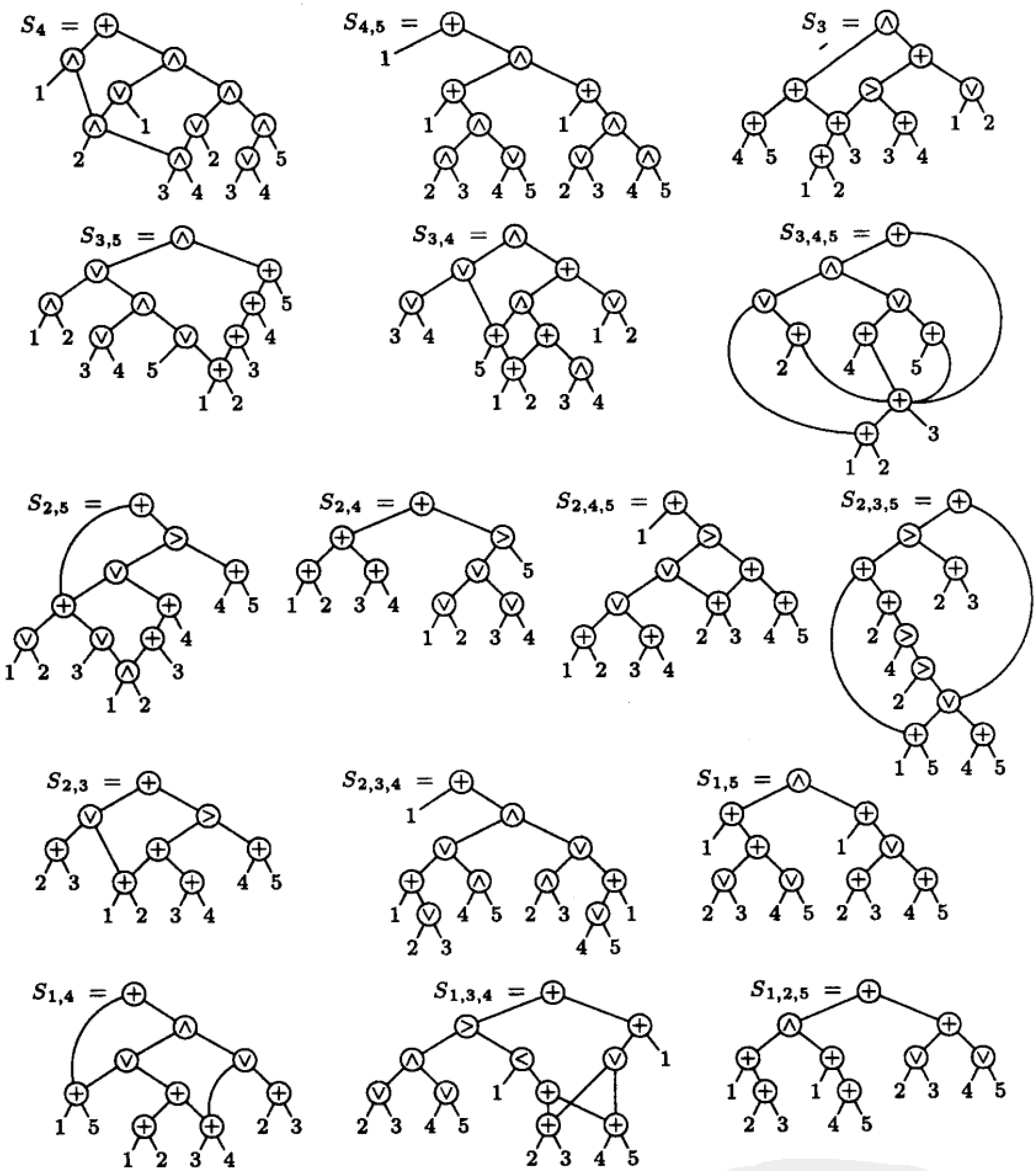


Fig. 10. Boolean chains of minimum cost for symmetric functions of five variables.

and others are simply astonishing. (Look, for example, at the 8-step computation of $S_{2,3}(x_1, x_2, x_3, x_4, x_5)$, or the elegant formula for $S_{2,3,4}$, or the nonmonotonic chains for $S_{4,5}$ and $S_{3,4,5}$.) Incidentally, Table 2 shows that all 5-variable functions have depth ≤ 4, but no attempt to minimize depth has been made in Fig. 10.

It turns out that all of these symmetric functions can be evaluated in minimum memory without increasing the cost. But no simple proof of that fact is known.

Multiple outputs. We often want to evaluate several different Boolean functions $f_1(x_1,\ldots,x_n)$, $\ldots$, $f_m(x_1,\ldots,x_n)$ at the same input values x_1, $\ldots$, x_n; in other words, we often want to evaluate a multibit function $y = f(x)$, where $y = f_1 \ldots f_m$ is a binary vector of length m and $x = x_1 \ldots x_n$ is a binary vector of length n. With luck, much of the work involved in the computation of one component value $f_j(x_1,\ldots,x_n)$ can be shared with the operations that are needed to evaluate the other component values $f_k(x_1,\ldots,x_n)$.

Let $C(f) = C(f_1 \ldots f_m)$ be the length of a shortest Boolean chain that computes all of the nontrivial functions f_j. More precisely, the chain $(x_{n+1},\ldots,x_{n+r})$ should have the property that, for $1 \le j \le m$, either $f_j(x_1,\ldots,x_n) = x_{l(j)}$ or $f_j(x_1,\ldots,x_n) = \bar{x}_{l(j)}$, for some $l(j)$ with $0 \le l(j) \le n+r$, where $x_0 = 0$. Clearly $C(f) \le C(f_1) + \cdots + C(f_m)$, but we might be able to do much better.

For example, suppose we want to compute the functions z_1 and z_0 defined by

$$(z_1 z_0)_2 \;=\; x_1 + x_2 + x_3, \qquad (21)$$

the two-bit binary sum of three Boolean variables. We have

$$z_1 \;=\; \langle x_1 x_2 x_3 \rangle \qquad \text{and} \qquad z_0 \;=\; x_1 \oplus x_2 \oplus x_3, \qquad (22)$$

so the individual costs are $C(z_1) = 4$ and $C(z_0) = 2$. But it's easy to see that the combined cost $C(z_1 z_0)$ is at most 5, because $x_1 \oplus x_2$ is a suitable first step in the evaluation of each bit z_j:

$$\begin{aligned} &x_4 = x_1 \oplus x_2, \quad z_0 = x_5 = x_3 \oplus x_4;\\ &x_6 = x_3 \wedge x_4, \quad x_7 = x_1 \wedge x_2, \quad z_1 = x_8 = x_6 \vee x_7. \end{aligned} \qquad (23)$$

Furthermore, exhaustive calculations show that $C(z_1 z_0) > 4$; hence $C(z_1 z_0) = 5$.

Electrical engineers traditionally call a circuit for (21) a *full adder*, because n such building blocks can be hooked together to add two n-bit numbers. The special case of (22) in which $x_3 = 0$ is also important, although it boils down simply to

$$z_1 \;=\; x_1 \wedge x_2 \qquad \text{and} \qquad z_0 \;=\; x_1 \oplus x_2 \qquad (24)$$

and has complexity 2; engineers call it a "half adder" in spite of the fact that the cost of a full adder exceeds the cost of two half adders.

The general problem of radix-2 addition

$$\begin{array}{r} (x_{n-1} \ldots x_1 x_0)_2 \\ (y_{n-1} \ldots y_1 y_0)_2 \\ \hline (z_n\, z_{n-1} \ldots z_1\, z_0)_2 \end{array} \qquad (25)$$

is to compute $n+1$ Boolean outputs $z_n \ldots z_1 z_0$ from the $2n$ Boolean inputs $x_{n-1} \ldots x_1 x_0 y_{n-1} \ldots y_1 y_0$; and it is readily solved by the formulas

$$c_{j+1} \;=\; \langle x_j y_j c_j \rangle, \qquad z_j \;=\; x_j \oplus y_j \oplus c_j, \qquad \text{for } 0 \le j < n, \qquad (26)$$

where the c_j are "carry bits" and we have $c_0 = 0$, $z_n = c_n$. Therefore we can use a half adder to compute c_1 and z_0, followed by $n-1$ full adders to compute the other c's and z's, accumulating a total cost of $5n-3$. And in fact N. P. Red'kin [*Problemy Kibernetiki* **38** (1981), 181–216] has proved that $5n-3$ steps

are actually necessary, by constructing an elaborate 35-page proof by induction, which concludes with Case 2.2.2.3.1.2.3.2.4.3(!). But the depth of this circuit, $2n-1$, is far too large for practical parallel computation, so a great deal of effort has gone into the task of devising circuits for addition that have depth $O(\log n)$ as well as reasonable cost. (See exercises 41–44.)

Now let's extend (21) and try to compute a general "sideways sum"

$$(z_{\lfloor \lg n \rfloor} \ldots z_1 z_0)_2 \;=\; x_1 + x_2 + \cdots + x_n. \qquad (27)$$

If $n = 2k+1$, we can use k full adders to reduce the sum to $(x_1 + \cdots + x_n) \bmod 2$ plus k bits of weight 2, because each full adder decreases the number of weight-1 bits by 2. For example, if $n = 9$ and $k = 4$ the computation is

$$\begin{array}{llll} x_{10}=x_1\oplus x_2\oplus x_3, & x_{11}=x_4\oplus x_5\oplus x_6, & x_{12}=x_7\oplus x_8\oplus x_9, & x_{13}=x_{10}\oplus x_{11}\oplus x_{12}, \\ y_1=\langle x_1x_2x_3\rangle, & y_2=\langle x_4x_5x_6\rangle, & y_3=\langle x_7x_8x_9\rangle, & y_4=\langle x_{10}x_{11}x_{12}\rangle, \end{array}$$

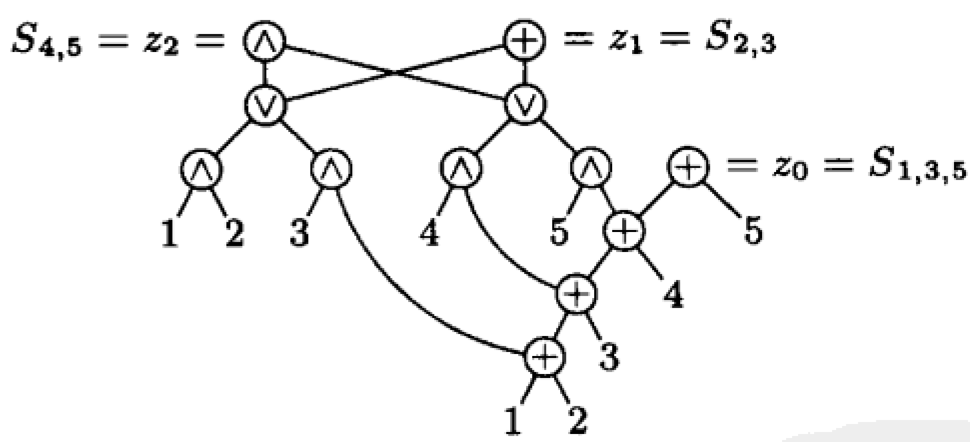
and we have $x_1 + \cdots + x_9 = x_{13} + 2(y_1 + y_2 + y_3 + y_4)$. If $n = 2k$ is even, a similar reduction applies but with a half adder at the end. The bits of weight 2 can then be summed in the same way; so we obtain the recurrence

$$s(n) \;=\; 5\lfloor n/2\rfloor - 3[n \text{ even}] + s(\lfloor n/2\rfloor), \qquad s(0)=0, \qquad (28)$$

for the total number of gates needed to compute $z_{\lfloor \lg n \rfloor} \ldots z_1 z_0$. (A closed formula for $s(n)$ appears in exercise 30.) We have $s(n) < 5n$, and the first values

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$s(n) =$	0	2	5	9	12	17	20	26	29	34	37	44	47	52	55	63	66	71	74	81

show that the method is quite efficient even for small n. For example, when $n = 5$ it produces



, (29)

which computes three different symmetric functions $z_2 = S_{4,5}(x_1, \ldots, x_5)$, $z_1 = S_{2,3}(x_1, \ldots, x_5)$, $z_0 = S_{1,3,5}(x_1, \ldots, x_5)$ in just 12 steps. The 10-step computation of $S_{4,5}$ is optimum, according to Fig. 10; of course the 4-step computation of $S_{1,3,5}$ is also optimum. Furthermore, although $C(S_{2,3}) = 8$, the function $S_{2,3}$ is computed here in a clever 10-step way that shares all but one gate with $S_{4,5}$.

Notice that we can now compute *any* symmetric function efficiently, because every symmetric function of $\{x_1, \ldots, x_n\}$ is a Boolean function of $z_{\lfloor \lg n \rfloor} \ldots z_1 z_0$. We know, for example, that any Boolean function of four variables has complexity ≤ 7; therefore any symmetric function $S_{k_1,\ldots,k_t}(x_1, \ldots, x_{15})$ costs at most $s(15) + 7 = 62$. Surprise: The symmetric functions of n variables were among the hardest of all to evaluate, when n was small, but they're among the easiest when $n \ge 10$.

We can also compute *sets* of symmetric functions efficiently. If we want, say, to evaluate all $n+1$ symmetric functions $S_k(x_1,\ldots,x_n)$ for $0 \le k \le n$ with a single Boolean chain, we simply need to evaluate the first $n+1$ *minterms* of z_0, z_1, $\ldots$, $z_{\lfloor \lg n \rfloor}$. For example, when $n = 5$ the minterms that give us all functions S_k are respectively $S_0 = \bar z_0 \wedge \bar z_1 \wedge \bar z_2$, $S_1 = \bar z_0 \wedge \bar z_1 \wedge z_2$, $\ldots$, $S_5 = z_0 \wedge \bar z_1 \wedge z_2$.

How hard is it to compute all 2^n minterms of n variables? Electrical engineers call this function an n-to-2^n *binary decoder*, because it converts n bits $x_1 \ldots x_n$ into a sequence of 2^n bits $d_0 d_1 \ldots d_{2^n-1}$, exactly one of which is 1. The principle of "divide and conquer" suggests that we first evaluate all minterms on the first $\lceil n/2 \rceil$ variables, as well as all minterms on the last $\lfloor n/2 \rfloor$; then 2^n AND gates will finish the job. The cost of this method is $t(n)$, where

$$t(0) = t(1) = 0; \qquad t(n) = 2^n + t(\lceil n/2 \rceil) + t(\lfloor n/2 \rfloor) \quad \text{for } n \ge 2. \tag{30}$$

So $t(n) = 2^n + O(2^{n/2})$; there's roughly one gate per minterm. (See exercise 32.)

Functions with multiple outputs often help us build larger functions with single outputs. For example, we've seen that the sideways adder (27) allows us to compute symmetric functions; and an n-to-2^n decoder also has many applications, in spite of the fact that 2^n can be huge when n is large. A case in point is the 2^m-*way multiplexer* $M_m(x_1,\ldots,x_m; y_0, y_1, \ldots, y_{2^m-1})$, also known as the m-bit *storage access function*, which has $n = m + 2^m$ inputs and takes the value y_k when $(x_1 \ldots x_m)_2 = k$. By definition we have

$$M_m(x_1,\ldots,x_m; y_0, y_1, \ldots, y_{2^m-1}) \;=\; \bigvee_{k=0}^{2^m-1} (d_k \wedge y_k), \tag{31}$$

where d_k is the kth output of an m-to-2^m binary decoder; thus, by (30), we can evaluate M_m with $2^m + (2^m-1) + t(m) = 3n + O(\sqrt{n}\,)$ gates. But exercise 39 shows that we can actually reduce the cost to only $2n + O(\sqrt{n}\,)$. (See also exercise 79.)

Asymptotic facts. When the number of variables is small, our exhaustive-search methods have turned up lots of cases where Boolean functions can be evaluated with stunning efficiency. So it's natural to expect that, when more variables are present, even more opportunities for ingenious evaluations will arise. But the truth is exactly the opposite, at least from a statistical standpoint:

Theorem S. *The cost of almost every Boolean function $f(x_1,\ldots,x_n)$ exceeds $2^n/n$. More precisely, if $c(n,r)$ Boolean functions have complexity $\le r$, we have*

$$(r-1)!\,c(n,r) \;\le\; 2^{2r+1}(n+r-1)^{2r}. \tag{32}$$

Proof. If a function can be computed in $r-1$ steps, it is also computable by an r-step chain. (This statement is obvious when $r = 1$; otherwise we can let $x_{n+r} = x_{n+r-1} \wedge x_{n+r-1}$.) We will show that there aren't very many r-step chains, hence we can't compute very many different functions with cost $\le r$.

Let π be a permutation of $\{1,\ldots,n+r\}$ that takes $1 \mapsto 1$, $\ldots$, $n \mapsto n$, and $n+r \mapsto n+r$; there are $(r-1)!$ such permutations. Suppose $(x_{n+1},\ldots,x_{n+r})$ is a

Boolean chain in which each of the intermediate steps $x_{n+1}, \ldots, x_{n+r-1}$ is used in at least one subsequent step. Then the permuted chains defined by the rule

$$x_i \;=\; x_{j'(i)} \circ'_i x_{k'(i)} \;=\; x_{j(i\pi)\pi^-} \circ_{i\pi} x_{k(i\pi)\pi^-}, \qquad \text{for } n < i \le n+r, \tag{33}$$

are distinct for different π. (If π takes $a \mapsto b$, we write $b = a\pi$ and $a = b\pi^-$.) For example, if π takes $5 \mapsto 6 \mapsto 7 \mapsto 8 \mapsto 9 \mapsto 5$, the chain (7) becomes

$$\begin{array}{cc}
\text{Original} & \text{Permuted} \\
x_5 = x_1 \oplus x_3, & x_5 = x_1 \oplus x_2, \\
x_6 = x_1 \oplus x_2, & x_6 = x_3 \oplus x_4, \\
x_7 = x_3 \oplus x_4, & x_7 = x_9 \vee x_5, \\
x_8 = x_5 \vee x_6, & x_8 = x_5 \oplus x_6, \\
x_9 = x_6 \oplus x_7, & x_9 = x_1 \oplus x_3, \\
x_{10} = x_8 \wedge \bar{x}_9; & x_{10} = x_7 \wedge \bar{x}_8.
\end{array} \tag{34}$$

Notice that we might have $j'(i) \ge k'(i)$ or $j'(i) > i$ or $k'(i) > i$, contrary to our usual rules. But the permuted chain computes the same function x_{n+r} as before, and it doesn't have any cycles by which an entry is defined indirectly in terms of itself, because the permuted x_i is the original $x_{i\pi}$.

We can restrict consideration to *normal* Boolean chains, as remarked earlier. So the $c(n,r)/2$ normal Boolean functions of cost $\le r$ lead to $(r-1)!\,c(n,r)/2$ different permuted chains, where the operator $\circ_i$ in each step is either $\wedge$, $\vee$, $\bar{\supset}$, or $\oplus$. And there are at most $4^r(n+r-1)^{2r}$ such chains, because there are four choices for $\circ_i$ and $n+r-1$ choices for each of $j(i)$ and $k(i)$, for $n < i \le n+r$. Equation (32) follows; and we obtain the opening statement of the theorem by setting $r = \lfloor 2^n/n \rfloor$. (See exercise 46.) ▮

On the other hand, there's also good news for infinity-minded people: We can actually evaluate every Boolean function of n variables with only slightly more than $2^n/n$ steps of computation, even if we avoid $\oplus$ and $\equiv$, using a technique devised by C. E. Shannon and improved by O. B. Lupanov [*Bell System Tech. J.* **28** (1949), 59–98, Theorem 6; *Isvestiia VUZov, Radiofizika* **1** (1958), 120–140].

In fact, the Shannon–Lupanov approach leads to useful results even when n is small, so let's get acquainted with it by studying a small example. Consider

$$f(x_1,x_2,x_3,x_4,x_5,x_6) \;=\; \bigl[(x_1x_2x_3x_4x_5x_6)_2 \text{ is prime}\bigr], \tag{35}$$

a function that identifies all 6-bit prime numbers. Its truth table has $2^6 = 64$ bits, and we can work with it conveniently by using a 4×16 array to look at those bits instead of confining ourselves to one dimension:

$$\begin{array}{r|cccccccccccccccc|l}
x_3 = & 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 & \\
x_4 = & 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 & \\
x_5 = & 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 & \\
x_6 = & 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 & \\
\hline
x_1x_2 = 00 & 0&0&1&1&0&1&0&1&0&0&0&1&0&1&0&0 & \rbrace\ \text{Group 1} \\
x_1x_2 = 01 & 0&1&0&1&0&0&0&1&0&0&0&0&0&1&0&1 & \\
x_1x_2 = 10 & 0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&1 & \rbrace\ \text{Group 2} \\
x_1x_2 = 11 & 0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0 & \\
\hline
\end{array} \tag{36}$$

The rows have been divided into two groups of two rows each; and each group of rows has 16 columns, which are of four basic types, namely $\genfrac{}{}{0pt}{}{0}{0}$, $\genfrac{}{}{0pt}{}{0}{1}$, $\genfrac{}{}{0pt}{}{1}{0}$, or $\genfrac{}{}{0pt}{}{1}{1}$. Thus we see that the function can be expressed as

$$\begin{aligned} f(x_1,\ldots,x_6) \;=\; & \bigl([x_1x_2 \in \{00\}] \wedge [x_3x_4x_5x_6 \in \{0010,0101,1011\}]\bigr) \\ \vee\; & \bigl([x_1x_2 \in \{01\}] \wedge [x_3x_4x_5x_6 \in \{0001,1111\}]\bigr) \\ \vee\; & \bigl([x_1x_2 \in \{00,01\}] \wedge [x_3x_4x_5x_6 \in \{0011,0111,1101\}]\bigr) \\ \vee\; & \bigl([x_1x_2 \in \{10\}] \wedge [x_3x_4x_5x_6 \in \{1001,1111\}]\bigr) \\ \vee\; & \bigl([x_1x_2 \in \{11\}] \wedge [x_3x_4x_5x_6 \in \{1101\}]\bigr) \\ \vee\; & \bigl([x_1x_2 \in \{10,11\}] \wedge [x_3x_4x_5x_6 \in \{0101,1011\}]\bigr). \end{aligned} \qquad (37)$$

(The first line corresponds to group 1, type $\genfrac{}{}{0pt}{}{1}{0}$, then comes group 1, type $\genfrac{}{}{0pt}{}{0}{1}$, etc.; the last line corresponds to group 2 and type $\genfrac{}{}{0pt}{}{1}{1}$.) A function like $[x_3x_4x_5x_6 \in \{0010,0101,1011\}]$ is the OR of three minterms of $\{x_3,x_4,x_5,x_6\}$.

In general we can view the truth table as a $2^k \times 2^{n-k}$ array, with l groups of rows having either $\lfloor 2^k/l \rfloor$ or $\lceil 2^k/l \rceil$ rows in each group. A group of size m will have columns of 2^m basic types. We form a conjunction $(g_{it}(x_1,\ldots,x_k) \wedge h_{it}(x_{k+1},\ldots,x_n))$ for each group i and each nonzero type t, where g_{it} is the OR of all minterms of $\{x_1,\ldots,x_k\}$ for the rows of the group where t has a 1, while h_{it} is the OR of all minterms of $\{x_{k+1},\ldots,x_n\}$ for the columns having type t in group i. The OR of all these conjunctions $(g_{it} \wedge h_{it})$ gives $f(x_1,\ldots,x_n)$.

Once we've chosen the parameters k and l, with $1 \le k \le n-2$ and $1 \le l \le 2^k$, the computation starts by computing all the minterms of $\{x_1,\ldots,x_k\}$ and all the minterms of $\{x_{k+1},\ldots,x_n\}$, in $t(k)+t(n-k)$ steps (see (30)). Then, for $1 \le i \le l$, we let group i consist of rows for the values of $(x_1,\ldots,x_k)$ such that $(i-1)2^k/l \le (x_1\ldots x_k)_2 < i2^k/l$; it contains $m_i = \lceil i2^k/l \rceil - \lceil (i-1)2^k/l \rceil$ rows. We form all functions g_{it} for $t \in S_i$, the family of $2^{m_i}-1$ nonempty subsets of those rows; $2^{m_i}-m_i-1$ ORs of previously computed minterms will accomplish that task. We also form all functions h_{it} representing columns of nonzero type t; for this purpose we'll need at most 2^{n-k} OR operations in each group i, since we can OR each minterm into the h function of the appropriate type t. Finally we compute $f = \bigvee_{i=1}^{l} \bigvee_{t \in S_i} (g_{it} \wedge h_{it})$; each AND operation is compensated by an unnecessary first OR into h_{it}. So the total cost is at most

$$t(k)+t(n-k)+(l-1)+\sum_{i=1}^{l}\bigl((2^{m_i}-m_i-1)+2^{n-k}+(2^{m_i}-2)\bigr); \qquad (38)$$

we want to choose k and l so that this upper bound is minimized. Exercise 52 discusses the best choice when n is small. And when n is large, a good choice yields a provably near-optimum chain, at least for most functions:

Theorem L. *Let $C(n)$ denote the cost of the most expensive Boolean functions of n variables. Then as $n \to \infty$ we have*

$$C(n) \;\ge\; \frac{2^n}{n}\Bigl(1+\frac{\lg n}{n}+O\Bigl(\frac{1}{n}\Bigr)\Bigr); \qquad (39)$$

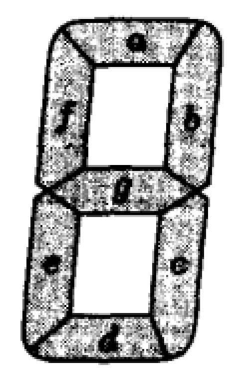
$$C(n) \;\le\; \frac{2^n}{n}\Bigl(1+3\frac{\lg n}{n}+O\Bigl(\frac{1}{n}\Bigr)\Bigr). \qquad (40)$$

Proof. Exercise 48 shows that the lower bound (39) is a consequence of Theorem S. For the upper bound, we set $k = \lfloor 2 \lg n \rfloor$ and $l = \lceil 2^k/(n - 3\lg n) \rceil$ in Lupanov's method; see exercise 53. ∎

Synthesizing a good chain. Formula (37) isn't the best way to implement a 6-bit prime detector, but it does suggest a decent strategy. For example, we needn't let variables x_1 and x_2 govern the rows: Exercise 51 shows that a better chain results if the rows are based on x_5x_6 while the columns come from $x_1x_2x_3x_4$, and in general there are many ways to partition a truth table by playing k of the variables against the other $n - k$.

Furthermore, we can improve on (37) by using our complete knowledge of all 4-variable functions; there's no need to evaluate a function like $[x_3x_4x_5x_6 \in \{0010, 0101, 1011\}]$ by first computing the minterms of $\{x_3, x_4, x_5, x_6\}$, if we know the best way to evaluate every such function from scratch. On the other hand, we do need to evaluate several 4-variable functions simultaneously, so the minterm approach might not be such a bad idea after all. Can we really improve on it?

Let's try to find a good way to synthesize a Boolean chain that computes a given set of 4-variable functions. The six functions of $x_3x_4x_5x_6$ in (37) are rather tame (see exercise 54), so we'll learn more by considering a more interesting example chosen from everyday life.



A *seven-segment display* is a now-ubiquitous way to represent a 4-bit number $(x_1x_2x_3x_4)_2$ in terms of seven cleverly positioned segments that are either visible or invisible. The segments are traditionally named (a, b, c, d, e, f, g) as shown; we get a '0' by turning on segments (a, b, c, d, e, f), but a '1' uses only segments (b, c). (Incidentally, the idea for such displays was invented by F. W. Wood, *U.S. Patent 974943* (1910), although Wood's original design used eight segments because he thought that a '4' requires a diagonal stroke.) Seven-segment displays usually support only the decimal digits '0', '1', ..., '9'; but of course a computer scientist's digital watch should display also hexadecimal digits. So we shall design seven-segment logic that displays the sixteen digits



(41)

when given the respective inputs $x_1x_2x_3x_4 = 0000, 0001, 0010, \ldots, 1111$.

In other words, we want to evaluate seven Boolean functions whose truth tables are respectively

$$\begin{aligned} a &= 1011\ 0111\ 1110\ 0011,\\ b &= 1111\ 1001\ 1110\ 0100,\\ c &= 1101\ 1111\ 1111\ 0100,\\ d &= 1011\ 0110\ 1101\ 1110,\\ e &= 1010\ 0010\ 1011\ 1111,\\ f &= 1000\ 1111\ 1111\ 0011,\\ g &= 0011\ 1110\ 1111\ 1111. \end{aligned} \tag{42}$$

If we simply wanted to evaluate each function separately, several methods that we've already discussed would tell us how to do it with minimum costs $C(a) = 5$, $C(b) = C(c) = C(d) = 6$, $C(e) = C(f) = 5$, and $C(g) = 4$; the total cost for all seven functions would then be 37. But we want to find a single Boolean chain that contains them all, and the shortest such chain is presumably much more efficient. How can we discover it?

Well, the task of finding a truly optimum chain for $\{a, b, c, d, e, f, g\}$ is probably infeasible from a computational standpoint. But a surprisingly good solution can be found with the help of the "footprint" idea explained earlier. Namely, we know how to compute not only a function's minimum cost, but also the set of all first steps consistent with that minimum cost in a normal chain. Function e, for example, has cost 5, but only if we evaluate it by starting with one of the instructions

$$x_5 = x_1 \oplus x_4 \qquad \text{or} \qquad x_5 = x_2 \wedge \bar{x}_3 \qquad \text{or} \qquad x_5 = x_2 \vee x_3.$$

Fortunately, one of the desirable first steps belongs to four of the seven footprints: Functions c, d, f, and g can all be evaluated optimally by starting with $x_5 = x_2 \oplus x_3$. So that is a natural choice; it essentially saves us three steps, because we know that at most 33 of the original 37 steps will be needed to finish.

Now we can recompute the costs and footprints of all 2^{16} functions, proceeding as before but also initializing the cost of the new function x_5 to zero. The costs of functions c, d, f, and g decrease by 1 as a result, and the footprints change too. For example, function a still has cost 5, but its footprint has increased from $\{x_1 \oplus x_3, x_2 \wedge x_3\}$ to $\{x_1 \oplus x_3, x_1 \wedge x_4, \bar{x}_1 \wedge x_4, x_2 \wedge x_3, \bar{x}_2 \wedge x_4, x_2 \oplus x_4, x_4 \wedge x_5, x_4 \oplus x_5\}$ when the function $x_5 = x_2 \oplus x_3$ is available for free.

In fact, $x_6 = \bar{x}_1 \wedge x_4$ is common to four of the new footprints, so again we have a natural way to proceed. And when everything is recalculated with zero cost given to both x_5 and x_6, the subsequent step $x_7 = x_3 \wedge \bar{x}_6$ turns out to be desirable in five of the newest footprints. Continuing in this "greedy" fashion, we aren't always so lucky, but a remarkable chain of only 22 steps does emerge:

$$\begin{array}{lll}
x_5 = x_2 \oplus x_3, & x_{13} = x_1 \oplus x_7, & \bar{a} = x_{20} = x_{14} \wedge \bar{x}_{19}, \\
x_6 = \bar{x}_1 \wedge x_4, & x_{14} = x_5 \oplus x_6, & \bar{b} = x_{21} = x_7 \oplus x_{12}, \\
x_7 = x_3 \wedge \bar{x}_6, & x_{15} = x_7 \vee x_{12}, & \bar{c} = x_{22} = \bar{x}_8 \wedge x_{15}, \\
x_8 = x_1 \oplus x_2, & x_{16} = x_1 \vee x_5, & \bar{d} = x_{23} = x_9 \wedge \bar{x}_{13}, \\
x_9 = x_4 \oplus x_5, & x_{17} = x_5 \vee x_6, & \bar{e} = x_{24} = x_6 \vee x_{18}, \\
x_{10} = \bar{x}_7 \wedge x_8, & x_{18} = x_9 \wedge x_{10}, & \bar{f} = x_{25} = \bar{x}_8 \wedge x_{17}, \\
x_{11} = x_9 \oplus x_{10}, & x_{19} = x_3 \wedge x_9, & g = x_{26} = x_7 \vee x_{16}. \\
x_{12} = x_5 \wedge x_{11}, & &
\end{array} \qquad (43)$$

(This is a *normal* chain, so it contains the normalizations $\{\bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{e}, \bar{f}, g\}$ instead of $\{a, b, c, d, e, f, g\}$. Simple changes will produce the unnormalized functions without changing the cost.)

Partial functions. In practice the output value of a Boolean function is often specified only at certain inputs $x_1 \ldots x_n$, and the outputs in other cases don't really matter. We might know, for example, that some of the input combinations

will never arise. In such cases, we place an asterisk into the corresponding positions of the truth table, instead of specifying 0 or 1 everywhere.

The seven-segment display provides a case in point, because most of its applications involve only the ten binary-coded decimal inputs for which we have $(x_1x_2x_3x_4)_2 \le 9$. We don't care what segments are visible in the other six cases. So the truth tables of (42) actually become

$$\begin{aligned} a &= \text{1011 0111 11** ****},\\ b &= \text{1111 1001 11** ****},\\ c &= \text{1101 1111 11** ****},\\ d &= \text{1011 0110 11** ****},\\ e &= \text{1010 0010 10** ****},\\ f &= \text{1000 111* 11** ****},\\ g &= \text{0011 1110 11** ****}. \end{aligned} \tag{44}$$

(Function f here has an asterisk also in position $x_1x_2x_3x_4 = 0111$, because a '7' can be displayed as either 7 or 7. Both of these styles appeared about equally often in the display units available to the author when this section was written. Truncated variants of the 6 and the 9 were sometimes seen in olden days, but they have thankfully disappeared.)

Asterisks in truth tables are generally known as *don't-cares* — a quaint term that could only have been invented by an electrical engineer. Table 3 shows that the freedom to choose arbitrary outputs is advantageous. For example, there are $\binom{16}{3}2^{13} = 4{,}587{,}520$ truth tables with 3 don't-cares; 69% of them cost 4 or less, even though only 21% of the asterisk-free truth tables permit such economy. On the other hand, don't-cares don't save us as much as we might hope; exercise 63 proves that a random function with, say, 30% don't-cares in its truth table tends to save only about 30% of the cost of a fully specified function.

What is the shortest Boolean chain that evaluates the seven partially specified functions in (44)? Our greedy-footprint method adapts itself readily to the presence of don't-cares, because we can OR together the footprints of all 2^d functions that match a pattern with d asterisks. The initial costs to evaluate each function separately are now reduced to $C(a) = 3$, $C(b) = C(c) = 2$, $C(d) = 5$, $C(e) = 2$, $C(f) = 3$, $C(g) = 4$, totalling just 21 instead of 37. Function g hasn't gotten cheaper, but it does have a larger footprint. Proceeding as before, but taking advantage of the don't-cares, we now can find a suitable chain of length only 13 — a chain with fewer than two operations per output(!):

$$\begin{array}{lll} x_5 = x_1 \oplus x_2, & \bar{e} = x_{10} = x_4 \vee x_8, & \bar{b} = x_{15} = x_2 \wedge \bar{x}_{13},\\ x_6 = x_3 \wedge \bar{x}_4, & g = x_{11} = x_7 \oplus x_8, & \bar{c} = x_{16} = \bar{x}_2 \wedge x_6,\\ x_7 = x_1 \oplus x_3, & x_{12} = x_4 \oplus x_{11}, & \bar{f} = x_{17} = \bar{x}_5 \wedge x_9.\\ x_8 = x_2 \wedge \bar{x}_6, & \bar{d} = x_{13} = x_{10} \wedge x_{12}, & \\ x_9 = x_3 \vee x_4, & \bar{a} = x_{14} = \bar{x}_3 \wedge x_{13}, & \end{array} \tag{45}$$

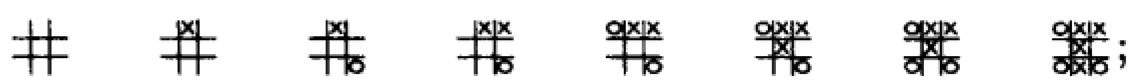
Tic-tac-toe. Let's turn now to a slightly larger problem, based on a popular children's game. Two players take turns filling the cells of a 3×3 grid. One player writes X's and the other writes O's, continuing until there either are three

Table 3

THE NUMBER OF 4-VARIABLE FUNCTIONS WITH d DON'T-CARES AND COST c

	$c=0$	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$	$c=7$
$d=0$	10	60	456	2474	10624	24184	25008	2720
$d=1$	160	960	7296	35040	131904	227296	119072	2560
$d=2$	1200	7200	52736	221840	700512	816448	166144	
$d=3$	5600	33600	228992	831232	2045952	1381952	60192	
$d=4$	18200	108816	666528	2034408	3505344	1118128	3296	
$d=5$	43680	257472	1367776	3351488	3491648	433568	32	
$d=6$	80080	455616	2015072	3648608	1914800	86016		
$d=7$	114400	606944	2115648	2474688	533568	12032		
$d=8$	128660	604756	1528808	960080	71520	896		
$d=9$	114080	440960	707488	197632	4160			
$d=10$	78960	224144	189248	20160				
$d=11$	41440	72064	25472	800				
$d=12$	15480	12360	1280					
$d=13$	3680	800						
$d=14$	480							
$d=15$	32							
$d=16$	1							

X's or three O's in a straight line (in which case that player wins) or all nine cells are filled without a winner (in which case it's a "cat's game" or tie). For example, the game might proceed thus:


(46)

X has won. Our goal is to design a machine that plays tic-tac-toe optimally—making a winning move from each position in which a forced victory is possible, and never making a losing move from a position in which defeat is avoidable.

More precisely, we will set things up so that there are 18 Boolean variables $x_1, \ldots, x_9, o_1, \ldots, o_9$, which govern lamps to illuminate cells of the current position. The cells are numbered $\begin{smallmatrix}1&2&3\\4&5&6\\7&8&9\end{smallmatrix}$ as on a telephone dial. Cell j displays an X if $x_j = 1$, an O if $o_j = 1$, or remains blank if $x_j = o_j = 0$.* We never have $x_j = o_j = 1$, because that would display '⊠'. We shall assume that the variables $x_1 \ldots x_9 o_1 \ldots o_9$ have been set to indicate a legal position in which nobody has won; the computer plays the X's, and it is the computer's turn to move. For this purpose we want to define nine functions $y_1, \ldots, y_9$, where y_j means "change x_j from 0 to 1." If the current position is a cat's game, we should make $y_1 = \cdots = y_9 = 0$; otherwise exactly one y_j should be equal to 1, and of course the output value $y_j = 1$ should occur only if $x_j = o_j = 0$.

With 18 variables, each of our nine functions y_j will have a truth table of size $2^{18} = 262{,}144$. It turns out that only 4520 legal inputs $x_1 \ldots x_9 o_1 \ldots o_9$ are

* This setup is based on an exhibit from the early 1950s at the Museum of Science and Industry in Chicago, where the author was first introduced to the magic of switching circuits. The machine in Chicago, designed circa 1940 by W. Keister at Bell Telephone Laboratories, allowed me to go first; yet I soon discovered that there was no way to defeat it. Therefore I decided to move as stupidly as possible, hoping that the designer had not anticipated such bizarre behavior. In fact I allowed the machine to reach a position where it had two winning moves; and it seized *both* of them! Moving twice is of course a flagrant violation of the rules, so I had won a moral victory even though the machine announced that I had lost.

I commenced an examination of a game called "tit-tat-to" ...
to ascertain what number of combinations were required
for all the possible variety of moves and situations.
I found this to be comparatively insignificant.
... A difficulty, however, arose of a novel kind.
When the automaton had to move, it might occur that there were
two different moves, each equally conducive to his winning the game.
... Unless, also, some provision were made,
the machine would attempt two contradictory motions.
— CHARLES BABBAGE, *Passages from the Life of a Philosopher* (1864)

possible, so those truth tables are 98.3% filled with don't-cares. Still, 4520 is uncomfortably large if we hope to design and understand a Boolean chain that makes sense intuitively. Section 7.1.4 will discuss alternative ways to represent Boolean functions, by which it is often possible to deal with hundreds of variables even though the associated truth tables are impossibly large.

Most functions of 18 variables require more than $2^{18}/18$ gates, but let's hope we can do better. Indeed, a plausible strategy for making suitable moves in tic-tac-toe suggests itself immediately, in terms of several conditions that aren't hard to recognize:

w_j, an X in cell j will win, completing a line of X's;
b_j, an O in cell j would lose, completing a line of O's;
f_j, an X in cell j will give X two ways to win;
d_j, an O in cell j would give O two ways to win.

For example, X's move to the center in (46) was needed to block O, so it was of type b_5; fortunately it was also of type f_5, forcing a win on the next move.

Let $L = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{1,4,7\},\{2,5,8\},\{3,6,9\},\{1,5,9\},\{3,5,7\}\}$ be the set of winning lines. Then we have

$$m_j = \bar{x}_j \wedge \bar{o}_j; \qquad \text{[moving in cell } j \text{ is legal]} \quad (47)$$
$$w_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(x_i \wedge x_k); \qquad \text{[moving in cell } j \text{ wins]} \quad (48)$$
$$b_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(o_i \wedge o_k); \qquad \text{[moving in cell } j \text{ blocks]} \quad (49)$$
$$f_j = m_j \wedge S_2\big(\{\alpha_{ik} \mid \{i,j,k\} \in L\}\big); \qquad \text{[moving in cell } j \text{ forks]} \quad (50)$$
$$d_j = m_j \wedge S_2\big(\{\beta_{ik} \mid \{i,j,k\} \in L\}\big); \qquad \text{[moving in cell } j \text{ defends]} \quad (51)$$

here α_{ik} and β_{ik} denote a single X or O together with a blank, namely

$$\alpha_{ik} = (x_i \wedge m_k) \vee (m_i \wedge x_k), \qquad \beta_{ik} = (o_i \wedge m_k) \vee (m_i \wedge o_k). \quad (52)$$

For example, $b_1 = m_1 \wedge \big((o_2 \wedge o_3) \vee (o_4 \wedge o_7) \vee (o_5 \wedge o_9)\big)$; $f_2 = m_2 \wedge S_2(\alpha_{13}, \alpha_{58}) = m_2 \wedge \alpha_{13} \wedge \alpha_{58}$; $d_5 = m_5 \wedge S_2(\beta_{19}, \beta_{28}, \beta_{37}, \beta_{46})$.

With these definitions we might try rank-ordering our moves thus:

$$\{w_1,\ldots,w_9\} > \{b_1,\ldots,b_9\} > \{f_1,\ldots,f_9\} > \{d_1,\ldots,d_9\} > \{m_1,\ldots,m_9\}. \quad (53)$$

"Win if you can; otherwise block if you can; otherwise fork if you can; otherwise defend if you can; otherwise make a legal move." Furthermore, when choosing

between legal moves it seems sensible to use the ordering

$$m_5 > m_1 > m_3 > m_9 > m_7 > m_2 > m_6 > m_8 > m_4, \qquad (54)$$

because 5, the middle cell, occurs in four winning lines, while a corner move to 1, 3, 9, or 7 occurs in three, and a side cell 2, 6, 8, or 4 occurs in only two. We might as well adopt this ordering of subscripts within all five groups of moves $\{w_j\}$, $\{b_j\}$, $\{f_j\}$, $\{d_j\}$, and $\{m_j\}$ in (53).

To ensure that at most one move is chosen, we define w'_j, b'_j, f'_j, d'_j, m'_j to mean "a prior choice is better." Thus, $w'_5 = 0$, $w'_1 = w_5$, $w'_3 = w_1 \vee w'_1$, $\ldots$, $w'_4 = w_8 \vee w'_8$, $b'_5 = w_4 \vee w'_4$, $b'_1 = b_5 \vee b'_5$, $\ldots$, $m'_4 = m_8 \vee m'_8$. Then we can complete the definition of a tic-tac-toe automaton by letting

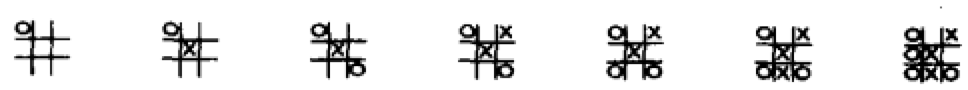
$$y_j = (w_j \wedge \bar{w}'_j) \vee (b_j \wedge \bar{b}'_j) \vee (f_j \wedge \bar{f}'_j) \vee (d_j \wedge \bar{d}'_j) \vee (m_j \wedge \bar{m}'_j), \quad \text{for } 1 \le j \le 9. \quad (55)$$

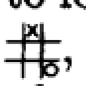
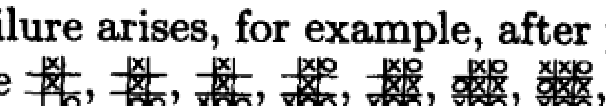
So we've constructed 9 gates for the m's, 48 for the w's, 48 for the b's, 144 for the α's and β's, 35 for the f's (with the help of Fig. 9), 35 for the d's, 43 for the primed variables, and 80 for the y's. Furthermore we can use our knowledge of partial 4-variable functions to reduce the six operations in (52) to only four,

$$\alpha_{ik} = (x_i \oplus x_k) \vee \overline{(o_i \oplus o_k)}, \qquad \beta_{ik} = \overline{(x_i \oplus x_k)} \vee (o_i \oplus o_k). \qquad (56)$$

This trick saves 48 gates; so our design has cost 396 gates altogether.

The strategy for tic-tac-toe in (47)–(56) works fine in most cases, but it also has some glaring glitches. For example, it loses ignominiously in the game

; (57)

the second × move is d_3, defending against a fork by ○, yet it actually forces ○ to fork in the opposite corner! Another failure arises, for example, after position , when move m_5 leads to the cat's game , instead of to the victory for × that appeared in (46). Exercise 65 patches things up and obtains a fully correct Boolean tic-tac-toe player that needs just 445 gates.

***Functional decomposition.** If the function $f(x_1, \ldots, x_n)$ can be written in the form $g(x_1, \ldots, x_k, h(x_{k+1}, \ldots, x_n))$, it's usually a good idea to evaluate $y = h(x_{k+1}, \ldots, x_n)$ first and then to compute $g(x_1, \ldots, x_k, y)$. Robert L. Ashenhurst inaugurated the study of such decompositions in 1952 [see *Annals Computation Lab. Harvard University* **29** (1957), 74–116], and observed that there's an easy way to recognize when f has this special property: If we write the truth table for f in a $2^k \times 2^{n-k}$ array as in (36), with rows for each setting of $x_1 \ldots x_k$ and columns for each setting of $x_{k+1} \ldots x_n$, then the desired subfunctions g and h exist if and only if the columns of this array have at most two different values. For example, the truth table for the function $\langle x_1 x_2 \langle x_3 x_4 x_5 \rangle\rangle$ is

0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	0	1	0	1	1	1
1	1	1	1	1	1	1	1

when expressed in this two-dimensional form. One type of column corresponds to the case $h(x_{k+1},\ldots,x_n)=0$; the other corresponds to $h(x_{k+1},\ldots,x_n)=1$.

In general the variables $X=\{x_1,\ldots,x_n\}$ might be partitioned into any two disjoint subsets $Y=\{y_1,\ldots,y_k\}$ and $Z=\{z_1,\ldots,z_{n-k}\}$, and we might have $f(x)=g(y,h(z))$. We could test for a (Y,Z) decomposition by looking at the columns of the $2^k\times 2^{n-k}$ truth table whose rows correspond to values of y. But there are 2^n such ways to partition X; and all of them are potential winners, except for trivial cases when $|Y|=0$ or $|Z|\le 1$. How can we avoid examining such a humungous number of possibilities?

A practical way to proceed was discovered by V. Y.-S. Shen, A. C. McKellar, and P. Weiner [*IEEE Transactions* **C-20** (1971), 304–309], whose method usually needs only $O(n^2)$ steps to identify any potentially useful partition (Y,Z) that may exist. The basic idea is simple: Suppose $x_i\in Z$, $x_j\in Z$, and $x_m\in Y$. Define eight binary vectors δ_l for $l=(l_1l_2l_3)_2$, where δ_l has (l_1,l_2,l_3) respectively in components (i,j,m), and zeros elsewhere. Consider any randomly chosen vector $x=x_1\ldots x_n$, and evaluate $f_l=f(x\oplus\delta_l)$ for $0\le l\le 7$. Then the four pairs

$$\binom{f_0}{f_1}\quad\binom{f_2}{f_3}\quad\binom{f_4}{f_5}\quad\binom{f_6}{f_7}\tag{58}$$

will appear in a 2×4 submatrix of the $2^k\times 2^{n-k}$ truth table. So a decomposition is impossible if these pairs are distinct, or if they contain three different values.

Let's call the pairs "good" if they're all equal, or if they have only two different values. Otherwise they're "bad." If f has essentially random behavior, we'll soon find bad pairs if we do this experiment with several different randomly chosen vectors x, because only 88 of the 256 possibilities for $f_0f_1\ldots f_7$ correspond to a good set of pairs; the probability of finding good pairs ten times in a row is only $(\frac{88}{256})^{10}\approx .00002$. And when we do discover bad pairs, we can conclude that

$$x_i\in Z\quad\text{and}\quad x_j\in Z\implies x_m\in Z,\tag{59}$$

because the alternative $x_m\in Y$ is impossible.

Suppose, for example, that $n=9$ and that f is the function whose truth table 11001001000011...00101 consists of the 512 most significant bits of π, in binary notation. (This is the "more-or-less random function" that we studied for $n=4$ in (5) and (6) above.) Bad pairs for this π function are quickly found in each of the cases (i,j,m) for which $m\ne i<j\ne m$. Indeed, in the author's experiments, 170 of those 252 cases were decided immediately; the average number of random x vectors per case was only 1.52; and only one case needed as many as eight x's before bad pairs appeared. Thus (59) holds for all relevant (i,j,m), and the function is clearly indecomposable. In fact, exercise 73 points out that we needn't make 252 tests to establish the indecomposability of this π function; only $\binom{n}{2}=36$ of them would have been sufficient.

Turning to a less random function, let $f(x_1,\ldots,x_9)=(\det X)\bmod 2$, where

$$X=\begin{pmatrix}x_1&x_2&x_3\\x_4&x_5&x_6\\x_7&x_8&x_9\end{pmatrix}.\tag{60}$$

This function does not satisfy condition (59) when $i = 1$, $j = 2$, and $m = 3$, because there are no bad pairs in that case. But it does satisfy (59) for $4 \le m \le 9$ when $\{i, j\} = \{1, 2\}$. We can denote this behavior by the convenient abbreviation '12⇒456789'; the full set of implications, for all pairs $\{i, j\}$, is

12⇒456789	18⇒34569	27⇒34569	37⇒24568	48⇒12369	67⇒12358
13⇒456789	19⇒24568	28⇒134679	38⇒14567	49⇒12358	68⇒12347
14⇒235689	23⇒456789	29⇒14567	39⇒124578	56⇒123789	69⇒124578
15⇒36789	24⇒36789	34⇒25789	45⇒123789	57⇒12369	78⇒123456
16⇒25789	25⇒134679	35⇒14789	46⇒123789	58⇒134679	79⇒123456
17⇒235689	26⇒14789	36⇒124578	47⇒235689	59⇒12347	89⇒123456

(see exercise 69). Bad pairs are a little more difficult to find when we probe this function at random: The average number of x's needed in the author's experiments rose to about 3.6, when bad pairs did exist. And of course there was a need to limit the testing, by choosing a tolerance threshold t and then giving up when t consecutive trials failed to find any bad pairs. Choosing $t = 10$ would have found all but 8 of the 198 implications listed above.

Implications like (59) are Horn clauses, and we know from Section 7.1.1 that it's easy to make further deductions from Horn clauses. Indeed, the method of exercise 74 will deduce that the only possible partition with $|Z| > 1$ is the trivial one ($Y = \emptyset$, $Z = \{x_1, \ldots, x_9\}$), after looking at fewer than 50 cases (i, j, m).

Similar results occur when $f(x_1, \ldots, x_9) = [\operatorname{per} X > 0]$, where per denotes the *permanent* function. (In this case f tells us if there is a matching in the bipartite subgraph of $K_{3,3}$ whose edges are specified by the variables $x_1 \ldots x_9$.) Now there are just 180 implications,

12⇒456789	18⇒3459	27⇒3459	37⇒2468	48⇒1269	67⇒1358
13⇒456789	19⇒2468	28⇒134679	38⇒1567	49⇒1358	68⇒2347
14⇒235689	23⇒456789	29⇒1567	39⇒124578	56⇒123789	69⇒124578
15⇒3678	24⇒3678	34⇒2579	45⇒123789	57⇒1269	78⇒123456
16⇒2579	25⇒134679	35⇒1489	46⇒123789	58⇒134679	79⇒123456
17⇒235689	26⇒1489	36⇒124578	47⇒235689	59⇒2347	89⇒123456,

only 122 of which would have been discovered with $t = 10$ as the cutoff threshold. (The best choice of t is not clear; perhaps it should vary dynamically.) Still, those 122 Horn clauses were more than enough to establish indecomposability.

What about a decomposable function? With $f = \langle x_2 x_3 x_6 x_9 \langle x_1 x_4 x_5 x_7 x_8 \rangle \rangle$ we get $i \wedge j \Rightarrow m$ for all $m \notin \{i, j\}$, except when $\{i, j\} \subseteq \{1, 4, 5, 7, 8\}$; in the latter case, m must also belong to $\{1, 4, 5, 7, 8\}$. Although only 185 of these 212 implications were discovered with tolerance $t = 10$, the partition $Y = \{x_2, x_3, x_6, x_9\}$, $Z = \{x_1, x_4, x_5, x_7, x_8\}$ emerged quickly as a strong possibility.

Whenever a potential decomposition is supported by the evidence, we need to verify that the corresponding $2^k \times 2^{n-k}$ truth table does indeed have only one or two distinct columns. But we're happy to spend 2^n units of time on that verification, because we've greatly simplified the evaluation of f.

The comparison function $f = \big[(x_1x_2x_3x_4)_2 \geq (x_5x_6x_7x_8)_2 + x_9\big]$ is another interesting case. Its 184 potentially deducible implications are

12⇒3456789	18⇒2345679	27⇒34689	37⇒489	48⇒9	67⇒23489
13⇒2456789	19⇒2345678	28⇒34679	38⇒479	49⇒8	68⇒23479
14⇒2356789	23⇒46789	29⇒34678	39⇒478	56⇒1234789	69⇒23478
15⇒2346789	24⇒36789	34⇒789	45⇒1236789	57⇒1234689	78⇒349
16⇒2345789	25⇒1346789	35⇒1246789	46⇒23789	58⇒1234679	79⇒348
17⇒2345689	26⇒34789	36⇒24789	47⇒389	59⇒1234678	89⇒4,

and 145 of them were found when $t = 10$. Three decompositions reveal themselves in this case, having $Z = \{x_4, x_8, x_9\}$, $Z = \{x_3, x_4, x_7, x_8, x_9\}$, and $Z = \{x_2, x_3, x_4, x_6, x_7, x_8, x_9\}$, respectively. Ashenhurst proved that we can reduce f immediately as soon as we find a nontrivial decomposition; the other decompositions will show up later, when we try to reduce the simpler functions g and h.

***Decomposition of partial functions.** When the function f is only partially specified, a decomposition with partition (Y, Z) hinges on being able to assign values to the don't-cares so that at most two different columns appear in the corresponding $2^k \times 2^{n-k}$ truth table.

Two vectors $u_1 \ldots u_m$ and $v_1 \ldots v_m$ consisting of 0s, 1s, and $*$s are said to be *incompatible* if either $u_j = 0$ and $v_j = 1$ or $u_j = 1$ and $v_j = 0$, for some j — equivalently, if the subcubes of the m-cube specified by u and v have no points in common. Consider the graph whose vertices are the columns of a truth table with don't-cares, where $u \!-\!\!-\! v$ if and only if u and v are incompatible. We can assign values to the $*$s to achieve at most two distinct columns if and only if this graph is *bipartite*. For if $u_1, \ldots, u_l$ are mutually compatible, their generalized consensus $u_1 \sqcup \cdots \sqcup u_l$, defined in exercise 7.1.1–32, is compatible with all of them. [See S. L. Hight, *IEEE Trans.* **C-22** (1973), 103–110; E. Boros, V. Gurvich, P. L. Hammer, T. Ibaraki, and A. Kogan, *Discrete Applied Math.* **62** (1995), 51–75.] Since a graph is bipartite if and only if it contains no odd cycles, we can easily test this condition with a depth-first search (see Section 7.4.1).

Consequently the method of Shen, McKellar, and Weiner works also when don't-cares are present: The four pairs in (58) are considered bad if and only if three of them are mutually incompatible. We can operate almost as before, although bad pairs will naturally be harder to find when there are lots of $*$s (see exercise 72). However, Ashenhurst's theorem no longer applies. When several decompositions exist, they all should be explored further, because they might use different settings of the don't-cares, and some might be better than the others.

Although most functions $f(x)$ have no simple decomposition $g(y, h(z))$, we needn't give up hope too quickly, because other forms like $g(y, h_1(z), h_2(z))$ might well lead to an efficient chain. If, for example, f is symmetric in three of its variables $\{z_1, z_2, z_3\}$, we can always write $f(x) = g\big(y, S_{1,2}(z_1, z_2, z_3), S_{1,3}(z_1, z_2, z_3)\big)$, since $S_{1,2}(z_1, z_2, z_3)$ and $S_{1,3}(z_1, z_2, z_3)$ characterize the value of $z_1 + z_2 + z_3$. (Notice that just four steps will suffice to compute both $S_{1,2}$ and $S_{1,3}$.)

In general, as observed by H. A. Curtis [*JACM* **8** (1961), 484–496], $f(x)$ can be expressed in the form $g(y, h_1(z), \ldots, h_r(z))$ if and only if the $2^k \times 2^{n-k}$ truth

table corresponding to Y and Z has at most 2^r different columns. And when don't-cares are present, the same result holds if and only if the incompatibility graph for Y and Z can be colored with at most 2^r colors.

For example, the function $f(x) = (\det X) \bmod 2$ considered above turns out to have eight distinct columns when $Z = \{x_4, x_5, x_6, x_7, x_8, x_9\}$; that's a surprisingly small number, considering that the truth table has 8 rows and 64 columns. From this fact we might be led to discover how to expand a determinant by cofactors of the first row,

$$f(x) \;=\; x_1 \wedge h_1(x_4, \ldots, x_9) \,\oplus\, x_2 \wedge h_2(x_4, \ldots, x_9) \,\oplus\, x_3 \wedge h_3(x_4, \ldots, x_9),$$

if we didn't already know such a rule.

When there are $d \le 2^r$ different columns, we can think of $f(x)$ as a function of y and $h(z)$, where h takes each binary vector $z_1 \ldots z_{n-k}$ into one of the values $\{0, 1, \ldots, d-1\}$. Thus $(h_1, \ldots, h_r)$ is essentially an encoding of the different column types, and we hope to find very simple functions $h_1, \ldots, h_r$ that provide such an encoding. Moreover, if d is strictly less than 2^r, the function $g(y, h_1, \ldots, h_r)$ will have many don't-cares that may well decrease its cost.

The distinct columns might also suggest a function g for which the h's have don't-cares. For example, we can use $g(y_1, y_2, h_1, h_2) = (y_1 \oplus (h_1 \wedge y_2)) \wedge h_2$ when all columns are either $(0,0,0,0)^T$ or $(0,0,1,1)^T$ or $(0,1,1,0)^T$; then the value of $h_1(z)$ is arbitrary when z corresponds to an all-zero column. H. A. Curtis has explained how to exploit this idea when $|Y| = 1$ and $|Z| = n-1$ [see *IEEE Transactions* **C-25** (1976), 1033–1044].

For a comprehensive discussion of decomposition techniques, see Richard M. Karp, *J. Society for Industrial and Applied Math.* **11** (1963), 291–335.

Larger values of n. We've been considering only rather tiny examples of Boolean functions. Theorem S tells us that large, random examples are inherently difficult; but practical examples might well be highly nonrandom. So it makes sense to search for simplifications using heuristic methods.

When n grows, the best ways currently known for dealing with Boolean functions generally start with a Boolean chain—not with a huge truth table—and they try to improve that chain via "local changes." The chain can be specified by a set of equations. Then, if an intermediate result is used in comparatively few subsequent steps, we can try to eliminate it, temporarily making those subsequent steps into functions of three variables, and reformulating those functions in order to make a better chain when possible.

For example, suppose the gate $x_i = x_j \circ x_k$ is used only once, in the gate $x_l = x_i \,\square\, x_m$, so that $x_l = (x_j \circ x_k) \,\square\, x_m$. Other gates might already exist, by which we have computed other functions of x_j, x_k, and x_m; and the definitions of x_j, x_k, and x_m may imply that some of the joint values of (x_j, x_k, x_m) are impossible. Thus we might be able to compute x_l from other gates by doing just one further operation. For example, if $x_i = x_j \wedge x_k$ and $x_l = x_i \vee x_m$, and if the values $x_j \vee x_m$ and $x_k \vee x_m$ appear elsewhere in the chain, we can set $x_l = (x_j \vee x_m) \wedge (x_k \vee x_m)$; this eliminates x_i and reduces the cost by 1. Or if,

say, $x_j \wedge (x_k \oplus x_m)$ appears elsewhere and we know that $x_j x_k x_m \neq 101$, we can set $x_l = x_m \oplus (x_j \wedge (x_k \oplus x_m))$.

If x_i is used only in x_l and x_l is used only in x_p, then gate x_p depends on four variables, and we might be able to reduce the cost by using our total knowledge of four-variable functions, obtaining x_p in a better way while eliminating x_i and x_l. Similarly, if x_i appears only in x_l and x_p, we can eliminate x_i if we find a better way to evaluate two different functions of four variables, possibly with don't-cares and with other functions of those four variables available for free. Again, we know how to solve such problems, using the footprint method discussed above.

When no local changes are able to decrease the cost, we can also try local changes that preserve or even increase the cost, in order to discover different kinds of chains that might simplify in other ways. We shall discuss such local search methods extensively in Section 7.10.

Excellent surveys of techniques for Boolean optimization, which electrical engineers call the problem of "multilevel logic synthesis," have been published by R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, *Proceedings of the IEEE* **78** (1990), 264–300, and in the book *Synthesis and Optimization of Digital Circuits* by G. De Micheli (McGraw–Hill, 1994).

Lower bounds. Theorem S tells us that nearly every Boolean function of $n \geq 12$ variables is hard to evaluate, requiring a chain whose length exceeds $2^n/n$. Yet modern computers, which are built from logic circuits involving electric signals that represent thousands of Boolean variables, happily evaluate zillions of Boolean functions every microsecond. Evidently there are plenty of important functions that can be evaluated quickly, in spite of Theorem S. Indeed, the proof of that theorem was indirect; we simply counted the cases of low cost, so we learned absolutely nothing about any particular examples that might arise in practice. When we want to compute a given function and we can only think of a laborious way to do the job, how can we be sure that there's no tricky shortcut?

The answer to that question is almost scandalous: After decades of concentrated research, computer scientists have been unable to find *any* explicit family of functions $f(x_1, \ldots, x_n)$ whose cost is inherently nonlinear, as n increases. The true behavior is $2^n/n$, but no lower bound as strong as $n \log\log\log n$ has yet been proved! Of course we could rig up artificial examples, such as "the lexicographically smallest truth table of length 2^n that isn't achievable by any Boolean chain of length $\lfloor 2^n/n \rfloor - 1$"; but such functions are surely not explicit. The truth table of an explicit function $f(x_1, \ldots, x_n)$ should be computable in at most, say, 2^{cn} units of time for some constant c; that is, the time needed to specify all of the function values should be polynomial in the length of the truth table. Under those ground rules, no family of single-output functions is currently known to have a combinational complexity that exceeds $3n + O(1)$ as $n \to \infty$. [See N. Blum, *Theoretical Computer Science* **28** (1984), 337–345.]

The picture is not totally bleak, because several interesting *linear* lower bounds have been proved for functions of practical importance. A basic way to obtain such results was introduced by N. P. Red'kin in 1970: Suppose we have

an optimum chain of cost r for $f(x_1,\ldots,x_n)$. By setting $x_n \leftarrow 0$ or $x_n \leftarrow 1$, we obtain reduced chains for the functions $g(x_1,\ldots,x_{n-1}) = f(x_1,\ldots,x_{n-1},0)$ and $h(x_1,\ldots,x_{n-1}) = f(x_1,\ldots,x_{n-1},1)$, having cost $r-u$ if x_n was used as an input to u different gates. Moreover, if x_n is used in a "canalizing" gate $x_i = x_n \circ x_k$, where the operator $\circ$ is neither $\oplus$ nor $\equiv$, some setting of x_n will force x_i to be constant, thereby further reducing the chain for g or h. Lower bounds on g and/or h therefore lead to a lower bound on f. (See exercises 77–81.)

But where are the proofs of nonlinear lower bounds? Almost every problem with a yes-no answer can be formulated as a Boolean function, so there's no shortage of explicit functions that we don't know how to evaluate in linear time, or even in polynomial time. For example, any directed graph G with vertices $\{v_1,\ldots,v_m\}$ can be represented by its adjacency matrix X, where $x_{ij} = [v_i \to v_j]$; then

$$f(x_{12},\ldots,x_{1m},\ldots,x_{m1},\ldots,x_{m(m-1)}) \;=\; [G \text{ has a Hamiltonian path}] \qquad (61)$$

is a Boolean function of $n = m(m-1)$ variables. We would dearly love to be able to evaluate this function in, say, n^4 steps. We do know how to compute the truth table for f in $O(m!\,2^n) = 2^{n+O(\sqrt{n}\,\log n)}$ steps, since only $m!$ potential Hamiltonian paths exist; thus f is indeed "explicit." But nobody knows how to evaluate f in polynomial time, or how to prove that there isn't a $4n$-step chain.

For all we know, short Boolean chains for f might exist, for each n. After all, Figs. 9 and 10 reveal the existence of fiendishly clever chains even in the cases of 4 and 5 variables. Efficient chains for all of the larger problems that we ever will need to solve might well be "out there" — yet totally beyond our grasp, because we don't have time to find them. Even if an omniscient being revealed the simple chains to us, we might find them incomprehensible, because the shortest proof of their correctness might be longer than the number of cells in our brains.

Theorem S rules out such a scenario for most Boolean functions. But fewer than 2^{100} Boolean functions will ever be of practical importance in the entire history of the world, and Theorem S tells us zilch about them.

In 1974, Larry Stockmeyer and Albert Meyer were, however, able to construct a Boolean function f whose complexity is provably huge. Their f isn't "explicit," in the precise sense described above, but it isn't artificial either; it arises naturally in mathematical logic. Consider symbolic statements such as

```
048+1015≠1063;                                         (62)
∀m∃n(m<n+1);                                           (63)
∀n∃m(m+1<n);                                           (64)
∀a∀b(b≥a+2⇒∃ab(a<ab∧ab<b));                            (65)
∀A∀B(A≡B⇔¬∃n(n∈A∧n∉B∨n∈B∧n∉A));                        (66)
∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≤n)));                       (67)
∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≥n)));                       (68)
∃P∀a((a∈P⇔a+3∉P)⇔a<1000);                              (69)
∀A∀B(∀C∀c(C≡A∧c=1∨C≡B∧c=0⇒(∀n(n∈C⇔n+1∈C)⇔c=1))⇒¬A≡B).  (70)
```

Stockmeyer and Meyer defined a language L by using the 63-character alphabet

`∀∃¬()≡∈∉+∧∨⇒⇔<≤=≠≥>abcdefghijklmnopqABCDEFGHIJKLMNOPQ0123456789`

and giving conventional meanings to these symbols. Strings of lowercase letters within the sentences of L, like '`ab`' in (65), represent numeric variables, restricted to nonnegative integers; strings of uppercase letters represent set variables, restricted to finite sets of such numbers. For example, (66) means, "For all finite sets A and B, we have $A = B$ if and only if there doesn't exist a number n that is in A but not in B, or in B but not in A." Some of these statements are true; others are false. (See exercise 82.)

All of the strings (62)–(70) belong to L, but the language is actually quite restricted: The only arithmetic operation allowed on a number is to add a constant; we can write '`a+13`' but not '`a+b`'. The only relation allowed between a number and a set is elementhood (∈ or ∉). The only relation allowed between sets is equality (≡). Furthermore all variables must be quantified by ∃ or ∀.*

Every sentence of L that has length $k \le n$ can be represented by a binary vector of length $6n$, with zeros in the last $6(n-k)$ bits. Let $f(x)$ be a Boolean function of $6n$ variables such that $f(x) = 1$ whenever x represents a true sentence of L, and $f(x) = 0$ whenever x represents a sentence that is false; the value of $f(x)$ is unspecified when x doesn't represent a meaningful sentence. The truth table for such a function f can be constructed in a finite number of steps, according to theorems of Büchi and Elgot [*Zeitschrift für math. Logik und Grundlagen der Math.* **6** (1960), 66–92; *Transactions of the Amer. Math. Soc.* **98** (1961), 21–51]. But "finite" does not mean "feasible": Stockmeyer and Meyer proved that

$$C(f) > 2^{r-5} \qquad \text{whenever } n \ge 460 + .302r + 5.08\ln r \text{ and } r > 36. \qquad (71)$$

In particular, we have $C(f) > 2^{426} > 10^{128}$ when $n = 621$. *A Boolean chain with that many gates could never be built*, since 10^{128} is a generous upper bound on the number of protons in the universe. So this is a fairly small, finite problem that will never be solved.

Details of Stockmeyer and Meyer's proof appear in *JACM* **49** (2002), 753–784. The basic idea is that the language L, though limited, is rich enough to describe truth tables and the complexity of Boolean chains, using fairly short sentences; hence f has to deal with inputs that essentially refer to themselves.

***For further reading.** Thousands of significant papers have been written about networks of Boolean gates, because such networks underlie so many aspects of theory and practice. We have focused in this section chiefly on topics that are relevant to computer programming for sequential machines. But other topics have also been extensively investigated, of primary relevance to parallel computation, such as the study of small-depth circuits in which gates can have any number of inputs ("unlimited fan-in"). Ingo Wegener's book *The Complexity of*

* Technically speaking, the sentences of L belong to "weak second-order monadic logic with one successor." Weak second-order logic allows quantification over finite sets; monadic logic with k successors is the theory of unlabeled k-ary trees.

Boolean Functions (Teubner and Wiley, 1987) provides a good introduction to the entire subject.

We have mostly considered Boolean chains in which all binary operators have equal importance. For our purposes, gates such as $\oplus$ or $\overline{\subset}$ are neither more nor less desirable than gates such as $\wedge$ or $\vee$. But it's natural to wonder if we can get by with only the monotone operators $\wedge$ and $\vee$ when we are computing a monotone function. Alexander Razborov has developed striking proof techniques to show that, in fact, monotone operators by themselves have inherently limited capabilities. He proved, for example, that all AND-OR chains to determine whether the permanent of an $n \times n$ matrix of 0s and 1s is zero or nonzero must have cost $n^{\Omega(\log n)}$. [See *Doklady Akademii Nauk SSSR* **281** (1985), 798–801; *Matematicheskie Zametki* **37** (1985), 887–900.] By contrast, we will see in Section 7.5.1 that this problem, equivalent to "bipartite matching," is solvable in only $O(n^{2.5})$ steps. Furthermore, the efficient methods in that section can be implemented as Boolean chains of only slightly larger cost, when we allow negation or other Boolean operations in addition to $\wedge$ and $\vee$. (Vaughan Pratt has called this "the power of negative thinking.") An introduction to Razborov's methods appears in exercises 85 and 86.

EXERCISES

1. [*24*] The "random" function in formula (6) corresponds to a Boolean chain of cost 4 and depth 4. Find a formula of depth 3 that has the same cost.

2. [*21*] Show how to compute (a) $w \oplus \langle xyz \rangle$ and (b) $w \wedge \langle xyz \rangle$ with formulas that have depth 3 and cost 5.

3. [*M23*] (B. I. Finikov, 1957.) If the Boolean function $f(x_1, \ldots, x_n)$ is true at exactly k points, prove that $L(f) < 2n + (k-2)2^{k-1}$. *Hint:* Think of $k = 3$ and $n = 10^6$.

4. [*M26*] (P. M. Spira, 1971.) Prove that the minimum depth and formula length of a Boolean function satisfy $\lg L(f) < D(f) \le \alpha \lg L(f) + 1$, where $\alpha = 2/\lg(\frac{3}{2}) \approx 3.419$. *Hint:* Every binary tree with $r \ge 3$ internal nodes contains a subtree with s internal nodes, where $\frac{1}{3}r \le s < \frac{2}{3}r$.

▸ **5.** [*21*] The Fibonacci threshold function $F_n(x_1, \ldots, x_n) = \langle x_1^{F_1} x_2^{F_2} \ldots x_{n-1}^{F_{n-1}} x_n^{F_{n-2}} \rangle$ was analyzed in exercise 7.1.1–101, when $n \ge 3$. Is there an efficient way to evaluate it?

6. [*20*] True or false: A Boolean function $f(x_1, \ldots, x_n)$ is normal if and only if it satisfies the general distributive law $f(x_1, \ldots, x_n) \wedge y = f(x_1 \wedge y, \ldots, x_n \wedge y)$.

7. [*20*] Convert the Boolean chain '$x_5 = x_1 \,\bar\vee\, x_4$, $x_6 = \bar{x}_2 \vee x_5$, $x_7 = \bar{x}_1 \wedge \bar{x}_3$, $x_8 = x_6 \equiv x_7$' to an equivalent chain $(\hat{x}_5, \hat{x}_6, \hat{x}_7, \hat{x}_8)$ in which every step is normal.

▸ **8.** [*20*] Explain why (11) is the truth table of variable x_k.

9. [*20*] Algorithm L determines the lengths of shortest formulas for all functions f, but it gives no further information. Extend the algorithm so that it also provides actual minimum-length formulas like (6).

▸ **10.** [*20*] Modify Algorithm L so that it computes $D(f)$ instead of $L(f)$.

▸ **11.** [*22*] Modify Algorithm L so that, instead of lengths $L(f)$, it computes upper bounds $U(f)$ and footprints $\phi(f)$ as described in the text.

12. [*15*] What Boolean chain is equivalent to the minimum-memory scheme (13)?

13. [*16*] What are the truth tables of f_1, f_2, f_3, f_4, and f_5 in example (13)?

14. [*22*] What's a convenient way to compute the $5n(n-1)$ truth tables of (17), given the truth table of g? (Use bitwise operations as in (15) and (16).)

15. [*28*] Find short-as-possible ways to evaluate the following functions using minimum memory: (a) $S_2(x_1, x_2, x_3, x_4)$; (b) $S_1(x_1, x_2, x_3, x_4)$; (c) the function in (18).

16. [*HM33*] Prove that fewer than 2^{118} of the 2^{128} Boolean functions $f(x_1, \ldots, x_7)$ are computable in minimum memory.

▸ **17.** [*25*] (M. S. Paterson, 1977.) Although Boolean functions $f(x_1, \ldots, x_n)$ cannot always be evaluated in n registers, prove that $n+1$ registers are always sufficient. In other words, show that there is always a sequence of operations like (13) to compute $f(x_1, \ldots, x_n)$ if we allow $0 \le j(i), k(i) \le n$.

▸ **18.** [*35*] Investigate optimum minimum-memory computations for $f(x_1, x_2, x_3, x_4, x_5)$: How many classes of five-variable functions have $C_m(f) = r$, for $r = 0, 1, 2, \ldots$?

19. [*M22*] If a Boolean chain uses n variables and has length $r < n+2$, prove that it must be either a "top-down" or a "bottom-up" construction.

▸ **20.** [*40*] (R. Schroeppel, 2004.) A Boolean chain is *canalizing* if it does not use the operators $\oplus$ or $\equiv$. Find the optimum cost, length, and depth of all 4-variable functions under this constraint. Does the footprint heuristic still give optimum results?

21. [*46*] For how many four-variable functions did the Harvard researchers discover an optimum vacuum-tube circuit in 1951?

22. [*21*] Explain the chain for S_3 in Fig. 10, by noting that it incorporates the chain for $S_{2,3}$ in Fig. 9. Find a similar chain for $S_2(x_1, x_2, x_3, x_4, x_5)$.

▸ **23.** [*23*] Figure 10 illustrates only 16 of the 64 symmetric functions on five elements. Explain how to write down optimum chains for the others.

24. [*47*] Does every symmetric function f have $C_m(f) = C(f)$?

▸ **25.** [*17*] Suppose we want a Boolean chain that includes *all* functions of n variables: Let $f_k(x_1, \ldots, x_n)$ be the function whose truth table is the binary representation of k, for $0 \le k < m = 2^{2^n}$. What is $C(f_0 f_1 \ldots f_{m-1})$?

26. [*25*] True or false: If $f(x_0, \ldots, x_n) = (x_0 \wedge g(x_1, \ldots, x_n)) \oplus h(x_1, \ldots, x_n)$, where g and h are nontrivial Boolean functions whose joint cost is $C(gh)$, then $C(f) = 2 + C(gh)$.

▸ **27.** [*23*] Can a full adder (22) be implemented in five steps using only minimum memory (that is, completely inside three one-bit registers)?

28. [*26*] Prove that $C(u'v') = C(u''v'') = 5$ for the two-output functions defined by

$$(u'v')_2 = (x + y - (uv)_2) \bmod 4, \qquad (u''v'')_2 = (-x - y - (uv)_2) \bmod 4.$$

Use these functions to evaluate $[(x_1 + \cdots + x_n) \bmod 4 = 0]$ in fewer than $2.5n$ steps.

29. [*M28*] Prove that the text's circuit for sideways addition (27) has depth $O(\log n)$.

30. [*M25*] Solve the binary recurrence (28) for the cost $s(n)$ of sideways addition.

31. [*21*] If $f(x_1, \ldots, x_n)$ is symmetric, prove that $C(f) \le 5n + O(n/\log n)$.

32. [*HM16*] Why does the solution to (30) satisfy $t(n) = 2^n + O(2^{n/2})$?

33. [*HM22*] True or false: If $1 \le N \le 2^n$, the first N minterms of $\{x_1, \ldots, x_n\}$ can all be evaluated in $N + O(\sqrt{N})$ steps, as $n \to \infty$ and $N \to \infty$.

▸ **34.** [*22*] A *priority encoder* has $n = 2^m - 1$ inputs $x_1 \ldots x_n$ and m outputs $y_1 \ldots y_m$, where $(y_1 \ldots y_m)_2 = k$ if and only if $k = \max\{j \mid j = 0 \text{ or } x_j = 1\}$. Design a priority encoder that has cost $O(n)$ and depth $O(m)$.

35. [*23*] If $n > 1$, show that the conjunctions $x_1 \wedge \cdots \wedge x_{k-1} \wedge x_{k+1} \wedge \cdots \wedge x_n$ for $1 \le k \le n$ can all be computed from $(x_1, \ldots, x_n)$ with total cost $\le 3n - 6$.

▸ **36.** [*M28*] (R. E. Ladner and M. J. Fischer, 1980.) Let y_k be the "prefix" $x_1 \wedge \cdots \wedge x_k$ for $1 \le k \le n$. Clearly $C(y_1 \ldots y_n) = n - 1$ and $D(y_1 \ldots y_n) = \lceil \lg n \rceil$; but we can't simultaneously minimize both cost and depth. Find a chain of optimum depth $\lceil \lg n \rceil$ that has cost $< 4n$.

37. [*M28*] (Marc Snir, 1986.) Given $n \ge m \ge 1$, consider the following algorithm:

S1. [Upward loop.] For $t \leftarrow 1, 2, \ldots, \lceil \lg m \rceil$, set $x_{\min(m,2^t k)} \leftarrow x_{2^t(k-1/2)} \wedge x_{\min(m,2^t k)}$ for $k \ge 1$ and $2^t(k - 1/2) < m$.

S2. [Downward loop.] For $t \leftarrow \lceil \lg m \rceil - 1, \lceil \lg m \rceil - 2, \ldots, 1$, set $x_{2^t(k+1/2)} \leftarrow x_{2^t k} \wedge x_{2^t(k+1/2)}$ for $k \ge 1$ and $2^t(k + 1/2) < m$.

S3. [Extension.] For $k \leftarrow m + 1, m + 2, \ldots, n$, set $x_k \leftarrow x_{k-1} \wedge x_k$. ▮

a) Prove that this algorithm solves the prefix problem of exercise 36: It transforms $(x_1, x_2, \ldots, x_n)$ into $(x_1, x_1 \wedge x_2, \ldots, x_1 \wedge x_2 \wedge \cdots \wedge x_n)$.

b) Let $c(m, n)$ and $d(m, n)$ be the cost and depth of the corresponding Boolean chain. Prove that, if n is sufficiently large, $c(m, n) + d(m, n) = 2n - 2$.

c) Given n, what is $d(n) = \min_{1 \le m \le n} d(m, n)$? Show that $d(n) < 2 \lg n$.

d) Prove that there's a Boolean chain of cost $2n - 2 - d$ and depth d for the prefix problem whenever $d(n) \le d < n$. (This cost is optimum, by exercise 81.)

38. [*25*] In Section 5.3.4 we studied *sorting networks*, by which $\hat{S}(n)$ comparator modules are able to sort n numbers $(x_1, x_2, \ldots, x_n)$ into ascending order. If the inputs x_j are 0s and 1s, each comparator module is equivalent to two gates $(x \wedge y, x \vee y)$; so a sorting network corresponds to a certain kind of Boolean chain, which evaluates n particular functions of $(x_1, x_2, \ldots, x_n)$.

a) What are the n functions $f_1 f_2 \ldots f_n$ that a sorting network computes?

b) Show that those functions $\{f_1, f_2, \ldots, f_n\}$ can be computed in $O(n)$ steps with a chain of depth $O(\log n)$. (Hence sorting networks aren't asymptotically optimal, Booleanwise.)

▸ **39.** [*M21*] (M. S. Paterson and P. Klein, 1980.) Implement the 2^m-way multiplexer $M_m(x_1, \ldots, x_m; y_0, y_1, \ldots, y_{2^m-1})$ of (31) with an efficient chain that simultaneously establishes the upper bounds $C(M_m) \le 2n + O(\sqrt{n})$ and $D(M_m) \le m + O(\log m)$.

40. [*25*] If $n \ge k \ge 1$, let $f_{nk}(x_1, \ldots, x_n)$ be the "k in a row" function,

$$(x_1 \wedge \cdots \wedge x_k) \vee (x_2 \wedge \cdots \wedge x_{k+1}) \vee \cdots \vee (x_{n+1-k} \wedge \cdots \wedge x_n).$$

Show that the cost $C(f_{nk})$ of this function is less than $4n - 3k$.

41. [*M23*] (*Conditional-sum adders.*) One way to accomplish binary addition (25) with depth $O(\log n)$ is based on the multiplexer trick of exercise 4: If $(xx')_2 + (yy')_2 = (zz')_2$, where $|x'| = |y'| = |z'|$, we have either $(x)_2 + (y)_2 = (z)_2$ and $(x')_2 + (y')_2 = (z')_2$, or $(x)_2 + (y)_2 + 1 = (z)_2$ and $(x')_2 + (y')_2 = (1z')_2$. To save time, we can compute *both* $(x)_2 + (y)_2$ and $(x)_2 + (y)_2 + 1$ simultaneously as we compute $(x')_2 + (y')_2$. Afterwards, when we know whether or not the less significant part $(x')_2 + (y')_2$ produces a carry, we can use multiplexers to select the correct bits for the most significant part.

If this method is used recursively to build $2n$-bit adders from n-bit adders, how many gates are needed when $n = 2^m$? What is the corresponding depth?

42. [*25*] In the binary addition (25), let $u_k = x_k \wedge y_k$ and $v_k = x_k \oplus y_k$ for $0 \le k < n$.

a) Show that $z_k = v_k \oplus c_k$, where the carry bits c_k satisfy

$$c_k = u_{k-1} \vee (v_{k-1} \wedge (u_{k-2} \vee (v_{k-2} \wedge (\cdots (u_1 \wedge v_0) \cdots)))).$$

b) Let $U_k^k = 0$, $V_k^k = 1$, and $U_j^{k+1} = u_k \vee (v_k \wedge U_j^k)$, $V_j^{k+1} = v_k \wedge V_j^k$, for $k \ge j$. Prove that $c_k = U_0^k$, and that $U_i^k = U_j^k \vee (V_j^k \wedge U_i^j)$, $V_i^k = V_j^k \wedge V_i^j$ for $i \le j \le k$.

c) Let $h(m) = 2^{m(m-1)/2}$. Show that when $n = h(m)$, the carries $c_1, \ldots, c_n$ can all be evaluated with depth $(m+1)m/2 \approx \lg n + \sqrt{2 \lg n}$ and with total cost $O(2^m n)$.

▸ **43.** [*28*] A *finite state transducer* is an abstract machine with a finite input alphabet A, a finite output alphabet B, and a finite set of internal states Q. One of those states, q_0, is called the "initial state." Given a string $\alpha = a_1 \ldots a_n$, where each $a_j \in A$, the machine computes a string $\beta = b_1 \ldots b_n$, where each $b_j \in B$, as follows:

T1. [Initialize.] Set $j \leftarrow 1$ and $q \leftarrow q_0$.

T2. [Done?] Terminate the algorithm if $j > n$.

T3. [Output b_j.] Set $b_j \leftarrow c(q, a_j)$.

T4. [Advance j.] Set $q \leftarrow d(q, a_j)$, $j \leftarrow j + 1$, and return to step T2. ▍

The machine has built-in instructions that specify $c(q, a) \in B$ and $d(q, a) \in Q$ for every state $q \in Q$ and every character $a \in A$. The purpose of this exercise is to show that, if the alphabets A and B of any finite state transducer are encoded in binary, the string β can be computed from α by a Boolean chain of size $O(n)$ and depth $O(\log n)$.

a) Consider the problem of changing a binary vector $a_1 \ldots a_n$ to $b_1 \ldots b_n$ by setting

$$b_j \leftarrow a_j \oplus [a_j = a_{j-1} = \cdots = a_{j-k} = 1 \text{ and } a_{j-k-1} = 0, \text{ where } k \text{ is odd}],$$

assuming that $a_0 = 0$. For example, $\alpha = 11001001000111111101101010 \mapsto \beta = 10001001000101010 01001010$. Prove that this transformation can be carried out by a finite state transducer with $|A| = |B| = |Q| = 2$.

b) Suppose a finite state transducer is in state q_j after reading $a_1 \ldots a_{j-1}$. Explain how to compute the sequence $q_1 \ldots q_n$ with a Boolean chain of cost $O(n)$ and depth $O(\log n)$, using the construction of Ladner and Fischer in exercise 36. (From this sequence $q_1 \ldots q_n$ it is easy to compute $b_1 \ldots b_n$, since $b_j = c(q_j, a_j)$.)

c) Apply the method of (b) to the problem in (a).

▸ **44.** [*26*] (R. E. Ladner and M. J. Fischer, 1980.) Show that the problem of binary addition (25) can be viewed as a finite state transduction. Describe the Boolean chain that results from the construction of exercise 43 when $n = 2^m$, and compare it to the conditional-sum adder of exercise 41.

45. [*HM20*] Why doesn't the proof of Theorem S simply argue that the number of ways to choose $j(i)$ and $k(i)$ so that $1 \le j(i), k(i) < i$ is $n^2(n+1)^2 \ldots (n+r-1)^2$?

▸ **46.** [*HM21*] Let $\alpha(n) = c(n, \lfloor 2^n/n \rfloor)/2^{2^n}$ be the fraction of n-variable Boolean functions $f(x_1, \ldots, x_n)$ for which $C(f) \le 2^n/n$. Prove that $\alpha(n) \to 0$ rapidly as $n \to \infty$.

47. [*M23*] Extend Theorem S to functions with n inputs and m outputs.

48. [*HM23*] Find the smallest integer $r = r(n)$ such that $(r-1)!\, 2^{2^n} \le 2^{2r+1}(n+r-1)^{2r}$, (a) exactly when $1 \le n \le 16$; (b) asymptotically when $n \to \infty$.

49. [*HM25*] Prove that, as $n \to \infty$, almost all Boolean functions $f(x_1, \ldots, x_n)$ have minimum formula length $L(f) > 2^n/\lg n - 2^{n+2}/(\lg n)^2$.

50. [*24*] What are the prime implicants and prime clauses of the prime-number function (35)? Express that function in (a) DNF (b) CNF of minimum length.

51. [*20*] What representation of the prime-number detector replaces (37), if rows of the truth table are based on x_5x_6 instead of x_1x_2?

52. [*23*] What choices of k and l minimize the upper bound (38) when $5 \le n \le 16$?

53. [*HM22*] Estimate (38) when $k = \lfloor 2\lg n \rfloor$ and $l = \lceil 2^k/(n - 3\lg n) \rceil$ and $n \to \infty$.

54. [*29*] Find a short Boolean chain to evaluate all six of the functions $f_j(x) = [x_1x_2x_3x_4 \in A_j]$, where $A_1 = \{0010, 0101, 1011\}$, $A_2 = \{0001, 1111\}$, $A_3 = \{0011, 0111, 1101\}$, $A_4 = \{1001, 1111\}$, $A_5 = \{1101\}$, $A_6 = \{0101, 1011\}$. (These six functions appear in the prime-number detector (37).) Compare your chain to the minterm-first evaluation scheme of Lupanov's general method.

55. [*34*] Show that the cost of the 6-bit prime-detecting function is at most 14.

▶ **56.** [*16*] Explain why all functions with 14 or more don't-cares in Table 3 have cost 0.

57. [*19*] What seven-segment "digits" are displayed when $(x_1x_2x_3x_4)_2 > 9$ in (45)?

▶ **58.** [*30*] A 4×4-bit *S-box* is a permutation of the 4-bit vectors $\{0000, 0001, \ldots, 1111\}$; such permutations are used as components of well-known cryptographic systems such as the Russian standard GOST 28147 (1989). Every 4×4-bit S-box corresponds to a sequence of four functions $f_1(x_1, x_2, x_3, x_4)$, ..., $f_4(x_1, x_2, x_3, x_4)$, which transform $x_1x_2x_3x_4 \mapsto f_1f_2f_3f_4$.

Find all 4×4-bit S-boxes for which $C(f_1) = C(f_2) = C(f_3) = C(f_4) = 7$.

59. [*29*] One of the S-boxes satisfying the conditions of exercise 58 takes $(0, \ldots, \mathtt{f}) \mapsto (\mathtt{0, 6, 5, b, 3, 9, f, e, c, 4, 7, 8, d, 2, a, 1})$; in other words, the truth tables of (f_1, f_2, f_3, f_4) are respectively (**179a**, **63e8**, **5b26**, **3e29**). Find a Boolean chain that evaluates these four "maximally difficult" functions in fewer than 20 steps.

60. [*23*] (Frank Ruskey.) Suppose $z = (x+y) \bmod 3$, where $x = (x_1x_2)_2$, $y = (y_1y_2)_2$, $z = (z_1z_2)_2$, and each two-bit value is required to be either 00, 01, or 10. Compute z_1 and z_2 from x_1, x_2, y_1, and y_2 in six Boolean steps.

61. [*34*] Continuing exercise 60, find a good way to compute $z = (x+y) \bmod 5$, using the three-bit values 000, 001, 010, 011, 100.

62. [*HM23*] Consider a random Boolean partial function of n variables that has $2^n c$ "cares" and $2^n d$ "don't-cares," where $c + d = 1$. Prove that the cost of almost all such partial functions exceeds $2^n c/n$.

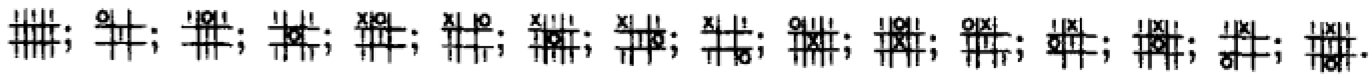
63. [*HM35*] (L. A. Sholomov, 1969.) Continuing exercise 62, prove that all such functions have cost $\le 2^n c/n(1 + O(n^{-1}\log n))$. *Hint:* There is a set of $2^m(1+k)$ vectors $x_1 \ldots x_k$ that intersects every $(k-m)$-dimensional subcube of the k-cube.

64. [*25*] (*Magic Fifteen.*) Two players alternately select digits from 1 to 9, using no digit twice; the winner, if any, is the first to get three digits that sum to 15. What's a good strategy for playing this game?

▶ **65.** [*35*] Modify the tic-tac-toe strategy of (47)–(56) so that it always plays correctly.

66. [*20*] Criticize the moves chosen in exercise 65. Are they always optimum?

▸ **67.** [*40*] Instead of simply finding one correct move for each position in tic-tac-toe, we might prefer to find them all. In other words, given $x_1 \ldots x_9 o_1 \ldots o_9$, we could try to compute nine outputs $g_1 \ldots g_9$, where $g_j = 1$ if and only if a move into cell j is among X's best. For example, exclamation marks indicate all of the right moves for X in the following typical positions:



A machine that chooses randomly among these possibilities is more fun to play against than a machine that has only one fixed strategy.

One attractive way to solve the all-good-moves problem is to use the fact that tic-tac-toe has eight symmetries. Imagine a chip that has 18 inputs $x_1 \ldots x_9 o_1 \ldots o_9$ and three outputs (c, s, m), for "corner," "side," and "middle," with the property that the desired functions g_j can be computed by hooking together eight of the chips appropriately:

$$\begin{aligned}
g_1 &= c(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\qquad \vee\, c(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9),\\
g_2 &= s(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\qquad \vee\, s(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7),\\
g_3 &= c(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7)\\
&\qquad \vee\, c(x_3x_6x_9x_2x_5x_8x_1x_4x_7o_3o_6o_9o_2o_5o_8o_1o_4o_7),\\
g_4 &= s(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9)\\
&\qquad \vee\, s(x_7x_4x_1x_8x_5x_2x_9x_6x_3o_7o_4o_1o_8o_5o_2o_9o_6o_3), \qquad \ldots\\
g_9 &= c(x_9x_8x_7x_6x_5x_4x_3x_2x_1o_9o_8o_7o_6o_5o_4o_3o_2o_1)\\
&\qquad \vee\, c(x_9x_6x_3x_8x_5x_2x_7x_4x_1o_9o_6o_3o_8o_5o_2o_7o_4o_1),
\end{aligned}$$

and g_5 is the OR of the m outputs from all eight chips.

Design such a chip, using fewer than 2000 gates.

68. [*M25*] Consider the n-bit π function $\pi_n(x_1 \ldots x_n)$, whose value is the $(x_1 \ldots x_n)_2$th bit to the right of the most significant bit in the binary representation of π. Does the method of exercise 4.3.1–39, which describes an efficient way to compute arbitrary bits of π, prove that $C(\pi_n) < 2^n/n$ for sufficiently large n?

69. [*M24*] Let the multilinear representation of f be

$$\alpha_{000} \oplus \alpha_{001}x_m \oplus \alpha_{010}x_j \oplus \alpha_{011}x_jx_m \oplus \alpha_{100}x_i \oplus \alpha_{101}x_ix_m \oplus \alpha_{110}x_ix_j \oplus \alpha_{111}x_ix_jx_m,$$

where each coefficient α_l is a function of the variables $\{x_1, \ldots, x_n\} \setminus \{x_i, x_j, x_m\}$.

a) Prove that the pairs (58) are "good" if and only if the coefficients satisfy

$$\alpha_{010}\alpha_{101} = \alpha_{011}\alpha_{100}, \quad \alpha_{101}\alpha_{110} = \alpha_{100}\alpha_{111}, \quad \text{and} \quad \alpha_{110}\alpha_{011} = \alpha_{111}\alpha_{010}.$$

b) For which values (i, j, m) are the pairs bad, when $f = (\det X) \bmod 2$? (See (60).)

▸ **70.** [*M27*] Let X be the 3×3 Boolean matrix (60). Find efficient chains for the Boolean functions (a) $(\det X) \bmod 2$; (b) $[\operatorname{per} X > 0]$; (c) $[\det X > 0]$.

▸ **71.** [*M26*] Suppose $f(x)$ is equal to 0 with probability p at each point $x = x_1 \ldots x_n$, independent of its value at other points.

a) What is the probability that the pairs (58) are good?
b) What is the probability that bad pairs (58) exist?
c) What is the probability that bad pairs (58) are found in at most t random trials?
d) What is the expected time to test case (i, j, m), as a function of p, t, and n?

72. [*M24*] Extend the previous exercise to the case of partial functions, where $f(x) = 0$ with probability p, $f(x) = 1$ with probability q, and $f(x) = *$ with probability r.

▶ **73.** [*20*] If bad pairs (58) exist for all (i, j, m) with $m \neq i \neq j \neq m$, show that the indecomposability of f can be deduced after testing only $\binom{n}{2}$ well-chosen triples (i, j, m).

74. [*25*] Extend the idea in the previous exercise, suggesting a strategy for choosing successive triples (i, j, m) when using the method of Shen, McKellar, and Weiner.

75. [*20*] What happens when the text's decomposition procedure is applied to the "all-equal" function $S_{0,n}(x_1, \ldots, x_n)$?

▶ **76.** [*M25*] (D. Uhlig, 1974.) The purpose of this exercise is to prove the amazing fact that, for certain functions f, the best chain to evaluate the Boolean function

$$F(u_1, \ldots, u_n, v_1, \ldots, v_n) = f(u_1, \ldots, u_n) \vee f(v_1, \ldots, v_n)$$

costs *less* than $2C(f)$; hence functional decomposition is *not* always a good idea.

We let $n = m + 2^m$ and write $f(i_1, \ldots, i_m, x_0, \ldots, x_{2^m-1}) = f_i(x)$, where i is regarded as the number $(i_1 \ldots i_m)_2$. Then $(u_1, \ldots, u_n) = (i_1, \ldots, i_m, x_0, \ldots, x_{2^m-1})$, $(v_1, \ldots, v_n) = (j_1, \ldots, j_m, y_0, \ldots, y_{2^m-1})$, and $F(u, v) = f_i(x) \vee f_j(y)$.

a) Prove that a chain of cost $O(n/\log n)^2$ suffices to evaluate the $2^m + 1$ functions

$$z_l = x \oplus (([l \le i] \oplus [i \le j]) \wedge (x \oplus y)), \qquad 0 \le l \le 2^m,$$

from given vectors i, j, x, and y; each z_l is a vector of length 2^m.

b) Let $g_i(x) = f_i(x) \oplus f_{i-1}(x)$ for $0 \le i \le 2^m$, where $f_{-1}(x) = f_{2^m}(x) = 0$. Estimate the cost of computing the $2^m + 1$ values $c_l = g_l(z_l)$, given the vectors z_l, for $0 \le l \le 2^m$.

c) Let $c'_l = c_l \wedge ([i \le j] \equiv [l \le i])$ and $c''_l = c_l \wedge ([i \le j] \equiv [j > l])$. Prove that

$$f_i(x) = c'_0 \oplus c'_1 \oplus \cdots \oplus c'_{2^m}, \qquad f_j(y) = c''_0 \oplus c''_1 \oplus \cdots \oplus c''_{2^m}.$$

d) Conclude that $C(F) \le 2^n/n + O(2^n(\log n)/n^2)$. (When n is sufficiently large, this cost is definitely less than $2^{n+1}/n$, but functions f exist with $C(f) > 2^n/n$.)

e) For clarity, write out the chain for F when $m = 1$ and $f(i, x_0, x_1) = (i \wedge x_0) \vee x_1$.

▶ **77.** [*35*] (N. P. Red'kin, 1970.) Suppose a Boolean chain uses only the operations AND, OR, or NOT; thus, every step is either $x_i = x_{j(i)} \wedge x_{k(i)}$ or $x_i = x_{j(i)} \vee x_{k(i)}$ or $x_i = \bar{x}_{j(i)}$. Prove that if such a chain computes either the "odd parity" function $f_n(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ or the "even parity" function $\bar{f}_n(x_1, \ldots, x_n) = 1 \oplus x_1 \oplus \cdots \oplus x_n$, where $n \ge 2$, the length of the chain is at least $4(n-1)$.

78. [*26*] (W. J. Paul, 1977.) Let $f(x_1, \ldots, x_m, y_0, \ldots, y_{2^m-1})$ be any Boolean function that equals y_k whenever $(x_1 \ldots x_m)_2 = k \in S$, for some given set $S \subseteq \{0, 1, \ldots, 2^m - 1\}$; we don't care about the value of f at other points. Show that $C(f) \ge 2\|S\| - 2$ whenever S is nonempty. (In particular, when $S = \{0, 1, \ldots, 2^m - 1\}$, the multiplexer chain of exercise 39 is asymptotically optimum.)

79. [*32*] (C. P. Schnorr, 1976.) Say that variables u and v are "mates" in a Boolean chain if there is exactly one simple path between them in the corresponding binary tree diagram. Two variables can be mates only if they are each used only once in the chain; but this necessary condition is not sufficient. For example, variables 2 and 4 are mates in the chain for $S_{1,2,3}$ in Fig. 9, but they are not mates in the chain for S_2.

a) Prove that a Boolean chain on n variables with no mates has cost $\ge 2n - 2$.

b) Prove that $C(f) = 2n - 3$ when f is the all-equal function $S_{0,n}(x_1, \ldots, x_n)$.

▶ **80.** [*M27*] (L. J. Stockmeyer, 1977.) Another notation for symmetric functions is sometimes convenient: If $\alpha = a_0a_1\ldots a_n$ is any binary string, let $S_\alpha(x) = a_{\nu x}$. For example, $\langle x_1x_2x_3\rangle = S_{0011}$ and $x_1 \oplus x_2 \oplus x_3 = S_{0101}$ in this notation. Notice that $S_\alpha(0, x_2, \ldots, x_n) = S_{\alpha'}(x_2, \ldots, x_n)$ and $S_\alpha(1, x_2, \ldots, x_n) = S_{'\alpha}(x_2, \ldots, x_n)$, where α' and $'\alpha$ stand respectively for α with its last or first element deleted. Also,

$$S_\alpha(f(x_3, \ldots, x_n), \bar f(x_3, \ldots, x_n), x_3, \ldots, x_n) = S_{'\alpha'}(x_3, \ldots, x_n)$$

when f is any Boolean function of $n-2$ variables.

a) A parity function has $a_0 \neq a_1 \neq a_2 \neq \cdots \neq a_n$. Assume that $n \geq 2$. Prove that if S_α is not a parity function and $S_{'\alpha'}$ isn't constant, then

$$C(S_\alpha) \geq \max(C(S_{\alpha'})+2, C(S_{'\alpha})+2, \min(C(S_{\alpha'})+3, C(S_{'\alpha})+3, C(S_{'\alpha'})+5)).$$

b) What lower bounds on $C(S_k)$ and $C(S_{\geq k})$ follow from this result, when $0 \leq k \leq n$?

81. [*23*] (M. Snir, 1986.) Show that any chain of cost c and depth d for the prefix problem of exercise 36 has $c + d \geq 2n - 2$.

▶ **82.** [*M23*] Explain the logical sentences (62)–(70). Which of them are true?

83. [*21*] If there's a Boolean chain for $f(x_1, \ldots, x_n)$ that contains p canalizing operations, show that $C(f) < (p+1)(n+p/2)$.

84. [*M20*] A *monotone Boolean chain* is a Boolean chain in which every operator $\circ_i$ is monotone. The length of a shortest monotone chain for f is denoted by $C^+(f)$. If there's a monotone Boolean chain for $f(x_1, \ldots, x_n)$ that contains p occurrences of $\wedge$ and q occurrences of $\vee$, show that $C^+(f) < \min((p+1)(n+p/2), (q+1)(n+q/2))$.

▶ **85.** [*M28*] Let M_n be the set of all monotone functions of n variables. If L is a family of functions contained in M_n, let

$$x \sqcup y = \bigwedge\{z \in L \mid z \supseteq x \vee y\} \qquad \text{and} \qquad x \sqcap y = \bigvee\{z \in L \mid z \subseteq x \wedge y\}.$$

We call L "legitimate" if it includes the constant functions 0 and 1 as well as the projection functions x_j for $1 \leq j \leq n$, and if $x \sqcup y \in L$, $x \sqcap y \in L$ whenever $x, y \in L$.

a) When $n = 3$ we can write $M_3 = \{$`00`, `01`, `03`, `05`, `11`, `07`, `13`, `15`, `0f`, `33`, `55`, `17`, `1f`, `37`, `57`, `3f`, `5f`, `77`, `7f`, `ff`$\}$, representing each function by its hexadecimal truth table. There are 2^{15} families L such that $\{$`00`, `0f`, `33`, `55`, `ff`$\} \subseteq L \subseteq M_3$; how many of them are legitimate?

b) If A is a subset of $\{1, \ldots, n\}$, let $\lceil A\rceil = \bigvee_{a\in A} x_a$; also let $\lceil\infty\rceil = 1$. Suppose $\mathcal{A}$ is a family of subsets of $\{1, \ldots, n\}$ that contains all sets of size ≤ 1 and is closed under intersection; in other words, $A \cap B \in \mathcal{A}$ whenever $A \in \mathcal{A}$ and $B \in \mathcal{A}$. Prove that the family $L = \{\lceil A\rceil \mid A \in \mathcal{A} \cup \{\infty\}\}$ is legitimate.

c) Let $(x_{n+1}, \ldots, x_{n+r})$ be a monotone Boolean chain (1). Suppose $(\hat x_{n+1}, \ldots, \hat x_{n+r})$ is obtained from the same Boolean chain, but with every operator $\wedge$ changed to $\sqcap$ and with every operator $\vee$ changed to $\sqcup$, with respect to some legitimate family L. Prove that, for $n+1 \leq l \leq n+r$, we must have

$$\hat x_l \subseteq x_l \vee \bigvee_{i=n+1}^{l} \{\hat x_i \oplus (\hat x_{j(i)} \vee \hat x_{k(i)}) \mid \circ_i = \vee\};$$

$$x_l \subseteq \hat x_l \vee \bigvee_{i=n+1}^{l} \{\hat x_i \oplus (\hat x_{j(i)} \wedge \hat x_{k(i)}) \mid \circ_i = \wedge\}.$$

86. [*HM37*] A graph G on vertices $\{1,\ldots,n\}$ can be defined by $N=\binom{n}{2}$ Boolean variables x_{uv} for $1\le u<v\le n$, where $x_{uv}=[u\text{---}v \text{ in } G]$. Let f be the function $f(x)=[G \text{ contains a triangle}]$; for example, when $n=4$, $f(x_{12},x_{13},x_{14},x_{23},x_{24},x_{34})=(x_{12}\wedge x_{13}\wedge x_{23})\vee(x_{12}\wedge x_{14}\wedge x_{24})\vee(x_{13}\wedge x_{14}\wedge x_{34})\vee(x_{23}\wedge x_{24}\wedge x_{34})$. The purpose of this exercise is to prove that the monotone complexity $C^+(f)$ is $\Omega(n/\log n)^3$.

a) If u_j — v_j for $1\le j\le r$ in a graph G, call $S=\{\{u_1,v_1\},\ldots,\{u_r,v_r\}\}$ an *r-family*, and let $\Delta(S)=\bigcup_{1\le i<j\le r}(\{u_i,v_i\}\cap\{u_j,v_j\})$ be the elements of its pairwise intersections. Say that G is *r-closed* if we have u — v whenever $\Delta(S)\subseteq\{u,v\}$ for some r-family S. It is *strongly* r-closed if, in addition, we have $|\Delta(S)|\ge 2$ for all r-families S. Prove that a strongly r-closed graph is also strongly $(r+1)$-closed.

b) Prove that the complete bigraph $K_{m,n}$ is strongly r-closed when $r>\max(m,n)$.

c) Prove that a strongly r-closed graph has at most $(r-1)^2$ edges.

d) Let L be the family of functions $\{1\}\cup\{\lceil G\rceil \mid G \text{ is a strongly } r\text{-closed graph on } \{1,\ldots,n\}\}$. (See exercise 85(b); we regard G as a set of edges. For example, when the edges are 1 — 3, 1 — 4, 2 — 3, 2 — 4, we have $\lceil G\rceil=x_{13}\vee x_{14}\vee x_{23}\vee x_{24}$.) Is L legitimate?

e) Let x_{N+1}, …, $x_{N+p+q}=f$ be a monotone Boolean chain with p $\wedge$-steps and q $\vee$-steps, and consider the modified chain $\hat{x}_{N+1}$, …, $\hat{x}_{N+p+q}=\hat{f}$ based on the family L in (d). If $\hat{f}\ne 1$, show that $2(r-1)^3p+(r-1)^2(n-2)\ge\binom{n}{3}$. *Hint:* Use the second formula in exercise 85(c).

f) Furthermore, if $\hat{f}=1$ we must have $r^2q\ge 2^{r-1}$.

g) Therefore $p=\Omega(n/\log n)^3$. *Hint:* Let $r\approx 6\lg n$ and apply exercise 84.

87. [*M20*] Show that when nonmonotonic operations are permitted, the triangle function of exercise 86 has cost $C(f)=O(n^{\lg 7}(\log n)^2)=O(n^{2.81})$. *Hint:* A graph has a triangle if and only if the cube of its adjacency matrix has a nonzero diagonal.

88. [*40*] A *median chain* is analogous to a Boolean chain, but it uses median-of-three steps $x_i=\langle x_{j(i)}x_{k(i)}x_{l(i)}\rangle$ for $n+1\le i\le n+r$, instead of the binary operations in (1).

Study the optimum length, depth, and cost of median chains, for all self-dual monotone Boolean functions of 7 variables. What is the shortest chain for $\langle x_1x_2x_3x_4x_5x_6x_7\rangle$?

ANSWERS TO EXERCISES

Answer not a fool according to his folly,
lest thou also be like unto him.
— Proverbs 26:4

SECTION 7

1. Following the hint, we'll want the second '$4m-4$' to be immediately followed by the first '$2m-1$'. The desired arrangements can be deduced from the first four examples, given in hexadecimal notation: **231213**, **46171435623725**, **86a31b1368597a425b2479**, **ca8e531f1358ac7db9e6427f2469bd**. [R. O. Davies, *Math. Gazette* **43** (1959), 253–255.]

2. Such arrangements exist if and only if $n \bmod 4 = 0$ or 1. This condition is necessary because there must be an even number of odd items. And it is sufficient because we can place '00' in front of the solutions in the previous exercise.

Notes: This question was first raised by Marshall Hall in 1951, and solved the following year by F. T. Leahy, Jr., in unpublished work [Armed Forces Security Agency report 343 (28 January 1952)]. It was independently posed and resolved by T. Skolem and T. Bang, *Math. Scandinavica* **5** (1957), 57–58. For other intervals of numbers, see the complete solution by J. E. Simpson, *Discrete Math.* **44** (1983), 97–104.

3. Yes. For example, the cycle (0072362435714165) can't be broken up.

4. The kth occurrence of b is in position $\lfloor k\phi \rfloor$ from the left, and the kth occurrence of a is in position $\lfloor k\phi^2 \rfloor$. Clearly $\lfloor k\phi^2 \rfloor - \lfloor k\phi \rfloor = k$, because $\phi^2 = \phi + 1$. (The integers $\lfloor k\phi \rfloor$ form the "spectrum" of ϕ; see exercise 3.13 of *CMath*.)

5. $2n-k-1$ of the $\binom{2n}{2}$ equally likely pairs of positions satisfy the stated condition. If these probabilities were independent (but they aren't), the value of $2L_n$ would be

$$\binom{2n}{2,2,\ldots,2}\prod_{k=1}^{n}\Bigl((2n-1-k)\big/\tbinom{2n}{2}\Bigr) = \frac{(2n)!^2\, n(n-1)}{n!\,(2n)^{n+1}(2n-1)^{n+1}}$$
$$= \exp\Bigl(n\ln\frac{4n}{e^3} + \ln\sqrt{\frac{\pi e n}{2}} + O(n^{-1})\Bigr).$$

6. (a) When the products are expanded, we obtain a polynomial of $(2n-2)!/(n-2)!$ terms, each of degree $4n$. There's a term $x_1^2 \ldots x_{2n}^2$ for each Langford pairing; every other term has at least one variable of degree 1. Summing over $x_1, \ldots, x_{2n} \in \{-1, +1\}$ therefore cancels out all the bad terms, but gives 2^{2n} for the good terms. An extra factor of 2 arises because there are $2L_n$ Langford pairings (including left-right reversals).

(b) Let $f_k = \sum_{j=1}^{2n-k-1} x_j x_{j+k+1}$ be the main part of the kth factor. We can run through all 4^n cases $x_1, \ldots, x_{2n} \in \{-1, +1\}$ in Gray-code order (Algorithm 7.2.1.1L), negating only one of the x_j each time. A change in x_j causes at most two adjustments to each f_k; so each Gray-code step costs $O(n)$.

We needn't compute the sum exactly; it suffices to work mod 2^N, where 2^N comfortably exceeds $2^{2n+1}L_n$. Even better, when $n = 24$, would be to do the computations mod $2^{60} - 1$, or mod both $2^{30} - 1$ and $2^{30} + 1$. One can also save $\lceil n/2 \rceil$ bits of precision by exploiting the fact that $f_k \equiv k + 1$ (modulo 2).

(c) The third equality is actually valid only when $n \bmod 4 = 0$ or 3; but those are the interesting n's. The sum can be carried out in n phases, where phase p for $p < n$ involves the cases where $x_{n-1} = x_{n+2}$, $x_{n-2} = x_{n+3}$, …, $x_{n-p+1} = x_{n+p}$, $x_{n-p} = x_n = x_{n+1} = +1$, and $x_{n+p+1} = -1$; it has an outer loop that chooses $(x_{n-p+1}, \ldots, x_{n-1})$ in all 2^{p-1} ways, and an inner loop that chooses $(x_1, \ldots, x_{n-p-1}, x_{n+p+2}, \ldots, x_{2n})$ in all $2^{2n-2p-2}$ ways. (The inner loop uses Gray binary code, preferably with "organ-pipe order" to prioritize the subscripts so that x_1 and x_{2n} vary most rapidly. The outer loop need not be especially efficient.) Phase n covers the 2^{n-1} palindromic cases with $x_j = x_{2n+1-j}$ for $1 \le j < n$ and $x_n = x_{n+1} = +1$. If s_p denotes the sum in phase p, then $s_1 + \cdots + s_{n-1} + \frac{1}{2}s_n = 2^{2n-2}L_n$.

A substantial fraction of the terms turn out be zero. For example, when $n = 16$, zeros appear about 76% of the time (in 408,838,754 cases out of $2^{29} + 2^{14}$). This fact can be used to avoid many multiplications in the inner loop. (Only f_1, f_3, … can be zero.)

7. Let d_k be the number of incomplete pairs after k characters have been read; thus $d_0 = d_{2n} = 0$, and $d_k = d_{k-1} \pm 1$ for $1 \le k \le 2n$. The largest such sequence in which d_k never exceeds 6 is $(0, 1, 2, 3, 4, 5, 6, 5, 6, \ldots, 5, 6, 5, 4, 3, 2, 1, 0)$. This sequence has $\sum_{k=1}^{2n} d_k = 11n - 30$. But $\sum_{k=1}^{2n} d_k = \sum_{k=1}^{n}(k+1) = \binom{n+1}{2} + n$ in any Langford pairing. Hence $\binom{n+1}{2} + n \le 11n - 30$, and $n \le 15$. (In fact, width 6 is also impossible when $n = 15$. The largest and smallest possible width are unknown in general.)

8. There are no solutions when $n = 4$ or $n = 7$. When $n = 8$ there are four:

1317538642572468; 1418634753268257; 4275248635713168; 5286235743681417.

(This problem makes a pleasant mechanical puzzle, using gadgets of width $k + 1$ and height $\lceil k/2 \rceil$ for piece k. In his original note [*Math. Gazette* **42** (1958), 228], C. Dudley Langford illustrated similar pieces, and exhibited a planar solution for $n = 12$. The question can be cast as an exact cover problem, with nonprimary columns representing places where two gadgets are not allowed to intersect; see exercise 7.2.2.1–00. Jean Brette has devised a somewhat similar puzzle, based on Skolem's variant of the problem and using width instead of planarity; he gave a copy to David Singmaster in 1992.)

9. Just three ways: 181915267285296475384639743, 191218246279458634753968357, 191618257269258476354938743 (and their reversals). [First found in 1969 by G. Baron; see *Combinatorial Theory and Its Applications* (Budapest: 1970), 81–92. The "dancing links" method of Section 7.2.2.1 resolves this question by traversing a search tree that has only 360 nodes, given an exact cover problem with 132 rows.]

10. For example, let A = 12, K = 8, Q = 4, J = 0, ♠ = 4, ♡ = 3, ♢ = 2, ♣ = 1; add.

[In this connection, orthogonal latin squares equivalent to Fig. 1 were implicitly present already in medieval Islamic talismans illustrated by Ibn al-Hajj in his *Kitab Shumus al-Anwar* (Cairo: 1322); he also gave a 5×5 example. See E. Doutté, *Magie*

et Religion dans l'Afrique du Nord (Algiers: 1909), 193–194, 214, 247; W. Ahrens, *Der Islam* **7** (1917), 228–238. See also an article on the history of latin squares being prepared by Lars D. Andersen.]

11. $\begin{pmatrix} d\gamma\text{א} & a\delta\text{ב} & b\beta\text{ג} & c\alpha\text{ד} \\ c\beta\text{ב} & b\alpha\text{א} & a\gamma\text{ד} & d\delta\text{ג} \\ a\alpha\text{ג} & d\beta\text{ד} & c\delta\text{א} & b\gamma\text{ב} \\ b\delta\text{ד} & c\gamma\text{ג} & d\alpha\text{ב} & a\beta\text{א} \end{pmatrix}$. [Joseph Sauveur presented the earliest known example of such squares in *Mémoires de l'Académie Royale des Sciences* (Paris, 1710), 92–138, §83.]

12. If n is odd, we can let $M_{ij} = (i - j) \bmod n$. But if n is even, there are no transversals: For if $\{(t_0{+}0) \bmod n, \ldots, (t_{n-1}{+}n{-}1) \bmod n\}$ is a transversal, we have $\sum_{k=0}^{n-1} t_k \equiv \sum_{k=0}^{n-1}(t_k + k)$ (modulo n), hence $\sum_{k=0}^{n-1} k = \frac{1}{2}n(n-1)$ is a multiple of n.

13. Replace each element l by $\lfloor l/5 \rfloor$ to get a matrix of 0s and 1s. Let the four quarters be named $\binom{A\ B}{C\ D}$; then A and D each contain exactly k 1s, while B and C each contain exactly k 0s. Suppose the original matrix has ten disjoint transversals. If $k \le 2$, at most four of them go through a 1 in A or D, and at most four go through a 0 in B or C. Thus at least two of them hit only 0s in A and D, only 1s in B and C. But such a transversal has an even number of 0s (not five), because it intersects A and D equally often.

Similarly, a latin square of order $4m + 2$ with an orthogonal mate must have more than m intruders in each of its $(2m + 1) \times (2m + 1)$ submatrices, under all renamings of the elements. [H. B. Mann, *Bull. Amer. Math. Soc.* (2) **50** (1944), 249–257.]

14. Cases (b) and (d) have no mates. Cases (a), (c), and (e) have respectively 2, 6, and 12265168(!), of which the lexicographically first and last are

(a)	(a)	(c)	(c)	(e)	(e)
0456987213	0691534782	0362498571	0986271435	0214365897	0987645321
1305629847	1308257964	1408327695	1354068792	1025973468	1795402638
2043798165	2169340578	2673519408	2741853960	2690587143	2506913874
3289176504	3250879416	3521970846	3572690814	3857694201	3154067289
4518263790	4587902631	4890253167	4630789251	4168730925	4231850967
5167432089,	5412763890;	5736841920,	5218947306;	5473829016,	5348276190.
6894015372	6945081327	6259784013	6095324178	6942158730	6820394715
7920341658	7836425109	7915602384	7869512043	7309216584	7069128543
8731504926	8723196045	8147036259	8407136529	8531402679	8412739056
9672850431	9074618253	9084165732	9123405687	9786041352	9673581402

Notes: Squares (a), (b), (c), and (d) were obtained from the decimal digits of π, e, γ, and ϕ, by discarding each digit that is inconsistent with a completed latin square. Although they aren't truly random, they're probably typical of 10×10 latin squares in general, roughly half of which appear to have orthogonal mates. Parker constructed square (e) in order to obtain an unusually large number of transversals; it has 5504 of them. (Euler had studied a similar example of order 6, therefore "just missing" the discovery of a 10×10 pair.)

15. Parker was dismayed to discover that none of the mates of square 14(e) are orthogonal to each other. With J. W. Brown and A. S. Hedayat [*J. Combinatorics, Inf. and System Sci.* **18** (1993), 113–115], he later found two $10{\times}10$s that have four disjoint common transversals (but not ten). [See also B. Ganter, R. Mathon, and A. Rosa, *Congressus Numerantium* **20** (1978), 383–398; **22** (1979), 181–204.] While pursuing an idea of L. Weisner [*Canadian Math. Bull.* **6** (1963), 61–63], the author accidentally noticed some squares that come even closer to a mutually orthogonal trio: The square below is orthogonal to its transpose; and it has five diagonally symmetric transversals, in cells $(0, p_0)$, …, $(9, p_9)$ for $p_0 \ldots p_9$ = 0132674598, 2301457689, 3210896745,

4897065312, and 6528410937, which are *almost* disjoint: They cover 49 cells.

$$L = \begin{pmatrix} 0234567891\\ 3192708546\\ 6528139407\\ 8753241960\\ 1689473025\\ 4970852613\\ 5047986132\\ 9416320758\\ 7361095284\\ 2805614379 \end{pmatrix} \perp \begin{pmatrix} 0368145972\\ 2157690438\\ 3925874160\\ 4283907615\\ 5712489306\\ 6034758291\\ 7891326054\\ 8549061723\\ 9406213587\\ 1670532849 \end{pmatrix} = L^T.$$

Extensive computations by B. D. McKay, A. Meynert, and W. Myrvold [*J. Comb. Designs* **15** (2007), 98–119] prove that no 10×10 latin square with nontrivial symmetry has two mates orthogonal to each other. Three mutually orthogonal latin squares are known to exist for all orders $n > 10$ [see S. M. P. Wang and R. M. Wilson, *Congressus Numerantium* **21** (1978), 688; D. T. Todorov, *Ars Combinatoria* **20** (1985), 45–47].

16. See R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory* (Cambridge University Press, 1991), §8.2.

17. (a) Let there be $3n$ columns r_j, c_j, v_j for $0 \le j < n$, and n^2 rows; row (i,j) has 1 in columns r_i, c_j, and v_l, where $l = L_{ij}$, for $0 \le i,j < n$.

(b) Let there be $4n^2$ columns r_{ij}, c_{ij}, x_{ij}, y_{ij} for $0 \le i,j < n$, and $n^3 - n^2 + n$ rows; row (i,j,k) has 1 in columns r_{ik}, c_{jk}, x_{ij}, and y_{lk}, where $l = L_{ij}$, for $0 \le i,j,k < n$ and ($i = k$ or $j > 0$).

18. Given an orthogonal array A with rows A_i for $1 \le i \le m$, define latin square $L_i = (L_{ijk})$ for $1 \le i \le m-2$ by setting $L_{ijk} = A_{iq}$ when $A_{(m-1)q} = j$ and $A_{mq} = k$, for $0 \le j,k < n$. (The value of q is uniquely determined by the values of j and k.) Permuting the columns of the array does not change the corresponding latin squares.

This construction can also be reversed, to produce orthogonal arrays of order n from mutually orthogonal latin squares of order n. In exercise 11, for example, we can let $a = \alpha = א = 0$, $b = \beta = ב = 1$, $c = \gamma = ג = 2$, and $d = \delta = ד = 3$, obtaining

$$A = \begin{pmatrix} 3012210303211230\\ 2310102301323201\\ 0123103223013210\\ 0000111122223333\\ 0123012301230123 \end{pmatrix}.$$

(The concept of an orthogonal array is mathematically "cleaner" than the concept of orthogonal latin squares, because it accounts better for the underlying symmetries. Notice, for example, that an $n \times n$ matrix L is a latin square if and only if it is orthogonal to two particular non-latin squares, namely

$$L \perp \begin{pmatrix} 1 & 1 & \ldots & 1\\ 2 & 2 & \ldots & 2\\ \vdots & \vdots & \ddots & \vdots\\ n & n & \ldots & n \end{pmatrix} \quad \text{and} \quad L \perp \begin{pmatrix} 1 & 2 & \ldots & n\\ 1 & 2 & \ldots & n\\ \vdots & \vdots & \ddots & \vdots\\ 1 & 2 & \ldots & n \end{pmatrix}.$$

Therefore Latin squares, Græco-Latin squares, Hebraic-Græco-Latin squares, etc., are equivalent to orthogonal arrays of depth 3, 4, 5, ... Moreover, the orthogonal arrays considered here are merely the special case $t = 2$ and $\lambda = 1$ of a more general concept of n-ary $m \times \lambda n^t$ arrays having "strength t" and "index λ," introduced by C. R. Rao

in *Proc. Edinburgh Math. Soc.* **8** (1949), 119–125; see the book *Orthogonal Arrays* by A. S. Hedayat, N. J. A. Sloane, and J. Stufken (Springer, 1999).)

19. We can rearrange the columns so that the first row is $0^n 1^n \ldots (n-1)^n$. Then we can renumber the elements of the other rows so that they begin with $01\ldots(n-1)$. The elements in each remaining column must then be distinct, in all rows but the first.

To achieve the upper bound when $n = p$, let each column be indexed by two numbers x and y, where $0 \le x, y < p$, and put the numbers y, x, $(x+y) \bmod p$, $(x+2y) \bmod p$, …, $(x+(p-1)y) \bmod p$ into that column. For example, when $p = 5$ we get the following orthogonal array, equivalent to four mutually orthogonal latin squares:

$$\begin{pmatrix} 0000011111222223333344444 \\ 0123401234012340123401234 \\ 0123412340234013401240123 \\ 0123423401401231234034012 \\ 0123434012123404012323401 \\ 0123440123340122340112340 \end{pmatrix}.$$

[Essentially the same idea works when n is a prime power, using the finite field $\mathrm{GF}(p^e)$; see E. H. Moore, *American Journal of Mathematics* **18** (1896), 264–303, §15(l). These arrays are equivalent to finite *projective planes*; see Marshall Hall, Jr., *Combinatorial Theory* (Blaisdell, 1967), Chapters 12 and 13.]

20. Let $\omega = e^{2\pi i/n}$, and suppose $a_1 \ldots a_{n^2}$ and $b_1 \ldots b_{n^2}$ are the vectors in different rows. Then $a_1 b_1 + \cdots + a_{n^2} b_{n^2} = \sum_{0\le j,k<n} \omega^{j+k} = 0$ because $\sum_{k=0}^{n-1} \omega^k = 0$.

21. (a) To show that equality-or-parallelism is an equivalence relation, we need to verify the transitive law: If $L \parallel M$ and $M \parallel N$ and $L \ne N$, then we must have $L \parallel N$. Otherwise there would be a point p with $L \cap N = \{p\}$, by (ii); and p would lie on two different lines parallel to M, contradicting (iii).

(b) Let $\{L_1, \ldots, L_n\}$ be a class of parallel lines, and assume that M is a line of another class. Then each L_j intersects M in a unique point p_j; and every point of M is encountered in this way, because every point of the geometry lies on exactly one line of each class, by (iii). Thus M contains exactly n points.

(c) We've already observed that every point belongs to m lines when there are m classes. If lines L, M, and N belong to three different classes, then M and N have the same number of points as the number of lines in L's class. So there's a common line size n, and in fact the total number of points is n^2. (Of course n might be infinite.)

22. Given an orthogonal array A of order n and depth m, define a geometric net with n^2 points and m classes of parallel lines by regarding the columns of A as points; line j of class k is the set of columns where symbol j appears in row k of A.

All finite geometric nets with $m \ge 3$ classes arise in this way. But a geometric net with only one class is trivially a partition of the points into disjoint subsets. A geometric net with $m = 2$ classes has nn' points (x, x'), where there are n lines '$x =$ constant' in one class and n' lines '$x' =$ constant' in the other. [For further information, see R. H. Bruck, *Canadian J. Math.* **3** (1951), 94–107; *Pacific J. Math.* **13** (1963), 421–457.]

23. (a) If $d(x,y) \le t$ and $d(x',y) \le t$ and $x \ne x'$, then $d(x,x') \le 2t$. Thus a code with distance $> 2t$ between codewords allows the correction of up to t errors — at least in principle, although the computations might be complex. Conversely, if $d(x,x') \le 2t$ and $x \ne x'$, there's an element y with $d(x,y) \le t$ and $d(x',y) \le t$; hence we can't reconstruct x uniquely when y is received.

(b, c) Let $m = r + 2$, and observe that a set of b^2 b-ary m-tuples has Hamming distance $\ge m - 1$ between all pairs of elements if and only if it forms the columns of a

b-ary orthogonal array of depth m. [See S. W. Golomb and E. C. Posner, *IEEE Trans.* **IT-10** (1964), 196–208. The literature of coding theory often denotes a code $C(b, n, r)$ of distance d by the symbol $(n + r, b^n, d)_b$. Thus, a b-ary orthogonal array of depth m is essentially an $(m, b^2, m - 1)_b$ code.]

24. (a) Suppose $x_j \neq x'_j$ for $1 \le j \le l$ and $x_j = x'_j$ for $l < j \le N$. We have $x = x'$ if $l = 0$. Otherwise consider the parity bits that correspond to the m lines through point 1. At most $l-1$ of those bits correspond to lines that touch the points $\{2, \ldots, l\}$. Hence x' has at least $m - (l-1)$ parity changes, and $d(x, x') \ge l + (m - (l-1)) = m + 1$.

(b) Let $l_{p1}, \ldots, l_{pm}$ be the index numbers of the lines through point p. After receiving a message $y_1 \ldots y_{N+R}$, compute x_p for $1 \le p \le N$ by taking the majority value of the $m + 1$ "witness bits" $\{y_{p0}, \ldots, y_{pm}\}$, where $y_{p0} = y_p$ and

$$y_{pk} = \bigl(y_{N+l_{pk}} + \textstyle\sum\{y_j \mid j \neq p \text{ and point } j \text{ lies on line } l_{pk}\}\bigr) \bmod 2, \quad \text{for } 1 \le k \le m.$$

This method works because each received bit y_j affects at most one of the witness bits.

For example, in the 25-point geometry of exercise 19, suppose the parity bit $x_{26+5i+j}$ of each codeword corresponds to line j of row i, for $0 \le i \le 5$ and $0 \le j < 5$; thus $x_{26} = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$, $x_{27} = x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}$, $\ldots$, $x_{55} = x_5 \oplus x_6 \oplus x_{12} \oplus x_{18} \oplus x_{24}$. Given message $y_1 \ldots y_{55}$, we decode bit x_1 (say) by computing the majority of the seven bits y_1, $y_{26} \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5$, $y_{31} \oplus y_6 \oplus y_{11} \oplus y_{16} \oplus y_{21}$, $y_{36} \oplus y_{10} \oplus y_{14} \oplus y_{18} \oplus y_{22}$, $y_{41} \oplus y_9 \oplus y_{12} \oplus y_{20} \oplus y_{23}$, $y_{46} \oplus y_8 \oplus y_{15} \oplus y_{17} \oplus y_{24}$, $y_{51} \oplus y_7 \oplus y_{13} \oplus y_{19} \oplus y_{25}$. [Section 7.1.2 explains how to calculate majority functions efficiently. Notice that we can eliminate the last 10 bits if we only wish to correct up to two errors, and the last 20 if single-error correction is sufficient. See M. Y. Hsiao, D. C. Bossen, and R. T. Chien, *IBM J. Research and Development* **14** (1970), 390–394.]

25. By considering anagrams of {`l`, `e`, `a`, `s`, `t`} (see exercise 5–21), we're led to the square

```
stela
telas
elast ,
laste
astel
```

and the cyclic rotations of its rows. Here `telas` are Spanish fabrics; `elast` is a prefix meaning flexible; and `laste` is an imperative Chaucerian verb. (Of course just about every pronounceable combination of five letters has been used to spell or misspell something somewhere, at some point in history.)

26. "`every night, young video buffs catch rerun fever forty years after those great shows first aired`." [Robert Leighton, *GAMES* **16**, 6 (December 1992), 34, 47.]

27. $(0, 4, 163, 1756, 3834)$ for $k = (1, 2, 3, 4, 5)$; `mamma` and `esses` give a "full house."

28. Yes, 38 pairs altogether. The "most common" solution is `needs` (rank 180) and `offer` (rank 384). Only three cases differ consistently by +1 (`adder beefs`, `sheer tiffs`, `sneer toffs`). Other memorable examples are `ghost hints` and `strut rusts`. One word of the pair ends with the letter `s` except in four cases, such as `robed spade`. [See Leonard J. Gordon, *Word Ways* **23** (1990), 59–61.]

29. There are 18 palindromes, from `level` (rank 184) to `dewed` (rank 5688). Some of the 34 mirror pairs are '`devil lived`', '`knits stink`', '`smart trams`', '`faced decaf`'.

30. Among 105 such words in the SGB, `first`, `below`, `floor`, `begin`, `cells`, `empty`, and `hills` are the most common; `abbey` and `pssst` are lexicographically first and last. (If you don't like `pssst`, the next-to-last is `mossy`.) Only 37 words, from `mecca` to `zoned`, have their letters in *reverse* order; but they are, of course, `wrong` answers.

31. The middle word is the average of the other two, so the extreme words must be congruent mod 2; this observation reduces the number of dictionary lookups by a factor of about 32. There are 119 such triples in WORDS(5757), but only two in WORDS(2000): marry, photo, solve; risky, tempo, vague. [*Word Ways* **25** (1992), 13–15.]

32. The only reasonably common example seems to be peopleless.

33. chief, fight, right, which, ouija, jokes, ankle, films, hymns, known, crops, pique, quart, first, first, study, mauve, vowel, waxes, proxy, crazy, pizza. (The idea is to find the most common word in which x is followed by $(x+1) \bmod 26$, for $x = \texttt{a}$ (0), $x = \texttt{b}$ (1), ..., $x = \texttt{z}$ (25). We also minimize the intervening distance, thus preferring bacon to the more common word black. In the one case where no such word exists, crazy seems most rational. See *OMNI* **16**, 8 (May 1994), 94.)

34. The top two (and total number) in each category are: pssst and pffft (2), schwa and schmo (2), threw and throw (36), three and spree (5), which and think (709), there and these (234), their and great (291), whooo and wheee (3), words and first (628), large and since (376), water and never (1313), value and radio (84), would and could (460), house and voice (101), quiet and queen (25), queue only (1), ahhhh and ankhs (4), angle and extra (20), other and after (227), agree and issue (20), along and using (124), above and alone (92), about and again (58), adieu and aquae (2), earth and eight (16), eagle and ounce (8), outer and eaten (42), eerie and audio (4), (0), ouija and aioli (2), (0), (0); years and every are the most common of the 868 omitted words. [To fill the three holes, Internet usage suggests ooops, ooooh, and ooooo. See P. M. Cohen, *Word Ways* **10** (1977), 221–223.]

35. Consider the collection WORDS(n) for $n = 1, 2, \ldots, 5757$. The illustrated trie, rooted at s, first becomes possible when n reaches 978 (the rank of stalk). The next root letter to support such a trie is c, which acquires enough branching in its descendants when $n = 2503$ (the rank of craze). Subsequent breakthroughs occur when $n = 2730$ (bulks), 3999 (ducky), 4230 (panty), 4459 (minis), 4709 (whooo), 4782 (lardy), 4824 (herem), 4840 (firma), 4924 (ridgy), 5343 (taxol).

(A breakthrough occurs when a top-level trie acquires Horton–Strahler number 4; see exercise 7.2.1.6–124. Amusing sets of words, suggestive of a new kind of poetry, arise also when the branching is right-to-left instead of left-to-right: black, slack, crack, track, click, slick, brick, trick, blank, plank, crank, drank, blink, clink, brink, drink. In fact, right-to-left branching yields a complete *ternary* trie with 81 leaves: males, sales, tales, files, miles, piles, holes, ..., tests, costs, hosts, posts.)

36. Denoting the elements of the cube by a_{ijk} for $1 \le i, j, k \le 5$, the symmetry condition is $a_{ijk} = a_{ikj} = a_{jik} = a_{jki} = a_{kij} = a_{kji}$. In general an $n \times n \times n$ cube has $3n^2$ words, obtained by fixing two coordinates and letting the third range from 1 to n; but the symmetry condition means that we need only $\binom{n+1}{2}$ words. Hence when $n = 5$ the number of necessary words is reduced from 75 to 15. [Jeff Grant was able to find 75 suitable words in the *Oxford English Dictionary*; see *Word Ways* **11** (1978), 156–157.]

Changing (stove, event) to (store, erect) or (stole, elect) gives two more.

37. The densest part of the graph, which we might call its "bare core," contains the vertices named bares and cores, which each have degree 25.

38. tears → raise → aisle → smile; the second word might also be reals. [Going from tears to smile as in (11) was one of Lewis Carroll's first five-letter examples. He would have been delighted to learn that the directed rule makes it more difficult to go from smile to tears, because *four* steps are needed in that direction.]

39. Always spanning, never induced.

40. (a) 2^e, (b) 2^n, one for each subset of E or V.

41. (a) $n = 1$ and $n = 2$; P_0 is undefined. (b) $n = 0$ and $n = 3$.

42. G has 65/2 edges (hence it doesn't exist).

43. Yes: The first three are isomorphic to Fig. 2(e). [The left-hand diagram is, in fact, identical to the earliest known appearance of the Petersen graph in print: See A. B. Kempe, *Philosophical Transactions* **177** (1886), 1–70, especially Fig. 13 in §59.] But the right-hand graph is definitely different; it is planar, Hamiltonian, and has girth 4.

44. Any automorphism must take a corner point into a corner point, because three distinct paths of length 2 can be found only between certain pairs of non-corner points. Therefore the graph has only the eight symmetries of C_4.

45. All edges of this graph connect vertices of the same row or adjacent rows. Therefore we can use the colors 0 and 2 alternately in even-numbered rows, 1 and 3 alternately in odd-numbered rows. The neighbors of `NV` form a 5-cycle, hence four colors are necessary.

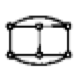
46. (a) Every vertex has degree ≥ 2, and its neighbors have a well-defined cyclic order corresponding to the incoming lines. If $u \mathbin{—} v$ and $u \mathbin{—} w$, where v and w are cyclically consecutive neighbors of u, we must have $v \mathbin{—} w$. Thus all points in the vicinity of any vertex u belong to a unique triangular region.

(b) The formula holds when $n = 3$. If $n > 3$, shrink any edge to a point; this transformation removes one vertex and three edges. (If $u \mathbin{—} v$ shrinks, suppose it was part of the triangles $x \mathbin{—} u \mathbin{—} v \mathbin{—} x$ and $y \mathbin{—} u \mathbin{—} v \mathbin{—} y$. We lose vertex v and edges $\{x \mathbin{—} v, u \mathbin{—} v, y \mathbin{—} v\}$; all other edges of the form $w \mathbin{—} v$ become $w \mathbin{—} u$.)

47. A planar diagram would divide the plane into regions, with either 4 or 6 vertices in the boundary of each region (because $K_{3,3}$ has no odd cycles). If there are f_4 and f_6 of each kind, we must have $4f_4 + 6f_6 = 18$, since there are 9 edges; hence $(f_4, f_6) = (3, 1)$ or $(0, 3)$. We could also triangulate the graph by adding $f_4 + 3f_6$ more edges; but then it would have at least 15 edges, contradicting exercise 46.

[The fact that $K_{3,3}$ is nonplanar goes back to a puzzle about connecting three houses to three utilities (water, gas, and electricity), without crossing pipes. Its origin is unknown; H. E. Dudeney called it "ancient" in *Strand* **46** (1913), 110.]

48. If u, v, w are vertices and $u \mathbin{—} v$, we must have $d(w, u) \not\equiv d(w, v)$ (modulo 2); otherwise shortest paths from w to u and from w to v would yield an odd cycle. After w is colored 0, the procedure therefore assigns the color $d(w, v) \bmod 2$ to each new uncolored vertex v that is adjacent to a colored vertex u; and every vertex v with $d(w, v) < \infty$ is colored before a new w is chosen.

49. There are only three: K_4, $K_{3,3}$, and  (which is $\overline{C}_6$).

50. The graph must be connected, because the number of 3-colorings is divisible by 3^r when there are r components. It must also be contained in a complete bipartite graph $K_{m,n}$, which can be 3-colored in $3(2^m + 2^n - 2)$ ways. Deleting edges from $K_{m,n}$ does not decrease the number of colorings; hence $2^m + 2^n - 2 \leq 8$, and we have $\{m, n\} = \{1, 1\}$, $\{1, 2\}$, $\{1, 3\}$, or $\{2, 2\}$. So the only possibilities are the claw $K_{1,3}$ and the path P_4.

51. A 4-cycle $p_1 \mathbin{—} L_1 \mathbin{—} p_2 \mathbin{—} L_2 \mathbin{—} p_1$ would correspond to two distinct lines $\{L_1, L_2\}$ with two common points $\{p_1, p_2\}$, contradicting (ii). So the girth is at least 6.

If there's only one class of parallel lines, the girth is ∞; if there are two classes, with $n \leq n'$ members, it is 8, or ∞ if $n = 1$. (See answer 22.) Otherwise we can find a 6-cycle by making a triangle from three lines that are chosen from different classes.

52. If the diameter is d and the girth is g, then $d \geq \lfloor g/2 \rfloor$, unless $g = \infty$.

53. **happy** (which is connected to **tears** and **sweat**, but not to **world**).

54. (a) It's a single, highly connected component. (Incidentally, this graph is the *line graph* of the bipartite graph in which one part corresponds to the initial letters {**A**, **C**, **D**, **F**, **G**, ..., **W**} and the other to the final letters {**A**, **C**, **D**, **E**, **H**, ..., **Z**}.)

(b) Vertex **WY** is isolated. The other vertices with in-degree zero, namely **FL**, **GA**, **PA**, **UT**, **WA**, **WI**, and **WV**, form strong components by themselves; they all precede a giant strong component, which is followed by each of the remaining single-vertex strong components with out-degree zero: **AZ**, **DE**, **KY**, **ME**, **NE**, **NH**, **NJ**, **NY**, **OH**, **TX**.

(c) Now the strong component {**GU**} precedes {**UT**}; **NH**, **OH**, **PA**, **WA**, **WI**, and **WV** join the giant strong component; {**FM**} precedes it; {**AE**} and {**WY**} follow it.

55. $\binom{N}{2} - \binom{n_1}{2} - \cdots - \binom{n_k}{2}$, where $N = n_1 + \cdots + n_k$.

56. True. Note that J_n is simple, but it doesn't correspond to any multigraph.

57. False, in the connected digraph $u \longrightarrow w \longleftarrow v$. (But u and v are in the same *strongly connected* component if and only if $d(u,v) < \infty$ and $d(v,u) < \infty$; see Section 2.3.4.2.)

58. Each component is a cycle whose order is at least (a) 3 (b) 1.

59. (a) By induction on n, we can use straight insertion sorting: Suppose $v_1 \longrightarrow \cdots \longrightarrow v_{n-1}$. Then either $v_n \longrightarrow v_1$ or $v_{n-1} \longrightarrow v_n$ or $v_{k-1} \longrightarrow v_n \longrightarrow v_k$, where k is minimum such that $v_n \longrightarrow v_k$. [L. Rédei, *Acta litterarum ac scientiarum* **7** (Szeged, 1934), 39–43.]

(b) 15: 01234, 02341, 02413, and their cyclic shifts. [The number of such oriented paths is always odd; see T. Szele, *Matematikai és Fizikai Lapok* **50** (1943), 223–256.]

(c) Yes. (By induction: If there's only one place to insert v_n as in part (a), the tournament is transitive.)

60. Set $A = \{x \mid u \longrightarrow x\}$, $B = \{x \mid x \longrightarrow v\}$, $C = \{x \mid v \longrightarrow x\}$. If $v \notin A$ and $A \cap B = \emptyset$ we have $|A| + |B| = |A \cup B| \leq n-2$, because $u \notin A \cup B$ and $v \notin A \cup B$. But $|B| + |C| = n-1$; hence $|A| < |C|$. [H. G. Landau, *Bull. Math. Biophysics* **15** (1953), 148.]

61. $1 \longrightarrow 1$, $1 \longrightarrow 2$, $2 \longrightarrow 2$; then $A = \binom{1\,1}{0\,1}$ and $A^k = \binom{1\,k}{0\,1}$ for all integers k.

62. (a) Suppose the vertices are $\{1, \ldots, n\}$. Each of the $n!$ terms $a_{1p_1} \ldots a_{np_n}$ in the expansion of the permanent is the number of spanning permutation digraphs that have arcs $j \longrightarrow p_j$. (b) A similar argument shows that $\det A$ is the number of even spanning permutation digraphs minus the number of odd ones. [See F. Harary, *SIAM Review* **4** (1962), 202–210, where permutation digraphs are called "linear subgraphs."]

63. Let v be any vertex. If $g = 2t+1$, at least $d(d-1)^{k-1}$ vertices x satisfy $d(v,x) = k$, for $1 \leq k < t$. If $g = 2t+2$ and v' is any neighbor of v, there also are at least $(d-1)^t$ vertices x for which $d(v,x) = t+1$ and $d(v',x) = t$.

64. To achieve the lower bound in answer 63, *every* vertex v must have degree d, and the d neighbors of v must all be adjacent to the remaining $d-1$ vertices. This graph is, in fact, $K_{d,d}$.

65. (a) By answer 63, G must be regular of degree d, and there must be exactly one path of length ≤ 2 between any two distinct vertices.

(b) We may take $\lambda_1 = d$, with $x_1 = (1 \ldots 1)^T$. All other eigenvectors satisfy $Jx_j = (0 \ldots 0)^T$; hence $\lambda_j^2 + \lambda_j = d-1$ for $1 < j \leq N$.

(c) If $\lambda_2 = \cdots = \lambda_m = (-1+\sqrt{4d-3})/2$ and $\lambda_{m+1} = \cdots = \lambda_N = (-1-\sqrt{4d-3})/2$, we must have $m-1 = N-m$. With this value we find $\lambda_1 + \cdots + \lambda_N = d - d^2/2$.

(d) If $4d-3=s^2$ and m is as in (c), the eigenvalues sum to

$$\frac{s^2+3}{4}+(m-1)\frac{s-1}{2}-\left(\frac{(s^2+3)^2}{16}+1-m\right)\frac{s+1}{2},$$

which is 15/32 plus a multiple of s. Hence s must be a divisor of 15.

[These results are due to A. J. Hoffman and R. R. Singleton, *IBM J. Research and Development* **4** (1960), 497–504, who also proved that the graph for $d=7$ is unique.]

66. Denote the 50 vertices by $[a,b]$ and (a,b) for $0 \le a,b < 5$, and define three kinds of edges, using arithmetic mod 5:

$$[a,b] — [a+1,b]; \qquad (a,b) — (a+2,b); \qquad (a,b) — [a+bc,c] \quad \text{for } 0 \le a,b,c < 5.$$

[See W. G. Brown, *Canadian J. Math.* **19** (1967), 644–648; *J. London Math. Soc.* **42** (1967), 514–520. Without the edges of the first two kinds, the graph has girth 6 and corresponds to a geometric net as in exercise 51, using the orthogonal array in answer 19.]

67. Certain possibilities have been ruled out by Michael Aschbacher in *Journal of Algebra* **19** (1971), 538–540.

68. If G has s automorphisms, it has $n!/s$ adjacency matrices, because there are s permutation matrices P such that $P^{-}AP=A$.

69. First set `IDEG(v)` $\leftarrow 0$ for all vertices v. Then perform (31) for all v, also setting $u \leftarrow$ `TIP(a)` and `IDEG(u)` $\leftarrow$ `IDEG(u)` $+1$ in the second line of that mini-algorithm.

To do something "for all v" using the SGB format, first set $v \leftarrow$ `VERTICES(g)`; then while $v <$ `VERTICES(g)` $+$ `N(g)`, do the operation and set $v \leftarrow v+1$.

70. Step B1 is performed once (but it takes $O(n)$ units of time). Steps (B2, B3, ..., B8) are performed respectively $(n+1, n, n, m+n, m, m, n)$ times, each with $O(1)$ cost.

71. Many choices are possible. Here we use 32-bit pointers, all relative to a symbolic address `Pool`, which lies in the `Data_Segment`. The following declarations provide one way to establish conventions for dealing with basic SGB data structures.

```
VSIZE IS 32 ;ASIZE IS 24                 Node sizes
ARCS IS 0 ;COLOR IS 8 ;LINK IS 12        Offsets of vertex fields
TIP IS 0 ;NEXT IS 4                      Offsets of arc fields
arcs GREG Pool+ARCS ;color GREG Pool+COLOR ;link GREG Pool+LINK
tip GREG Pool+TIP ;next GREG Pool+NEXT
u GREG ;v GREG ;w GREG ;s GREG ;a GREG ;mone GREG -1
AlgB   BZ     n,Success                  Exit if the graph is null.
       MUL    $0,n,VSIZE                 B1. Initialize.
       ADDU   v,v0,$0                    v ← v0 + n.
       SET    w,v0                       w ← v0.
1H     STT    mone,color,w               COLOR(w) ← −1.
       ADDU   w,w,VSIZE                  w ← w + 1.
       CMP    $0,w,v
       PBNZ   $0,1B                      Repeat until w = v.
0H     SUBU   w,w,VSIZE                  w ← w − 1.
3H     LDT    $0,color,w                 B3. Color w if necessary.
       PBNN   $0,2F                      To B2 if COLOR(w) ≥ 0.
       STCO   0,link,w                   COLOR(w) ← 0, LINK(w) ← Λ.
       SET    s,w                        s ← w.
4H     SET    u,s                        B4. Stack ⇒ u. Set u ← s.
```

```
         LDTU  s,link,s       s ← LINK(s).
         LDT   $1;color,u
         NEG   $1,1,$1        $1 ← 1 − COLOR(u).
         LDTU  a,arcs,u       a ← ARCS(u).
5H       BZ    a,8F           B5. Done with u? To B8 if a = Λ.
5H       LDTU  v,tip,a        v ← TIP(a).
6H       LDT   $0,color,v     B6. Process v.
         CMP   $2,$0,$1       (Here the program is slightly clever)
         PBZ   $2,7F          To B7 if COLOR(v) = 1 − COLOR(u).
         BNN   $0,Failure     Fail if COLOR(v) = COLOR(u).
         STT   $1,color,v     COLOR(v) ← 1 − COLOR(u).
         STTU  s,link,v       LINK(v) ← s.
         SET   s,v            s ← v.
7H       LDTU  a,next,a       B7. Loop on a. Set a ← NEXT(a).
         PBNZ  a,5B           To B5 if a ≠ Λ.
8H       PBNZ  s,4B           B8. Stack nonempty? To B4 if s ≠ Λ.
2H       CMP   $0,w,v0        B2. Done?
         PBNZ  $0,0B          If w ≠ v0, decrease w and go to B3.
Success  LOC   @              (Successful termination) ▮
```

72. (a) This condition clearly remains invariant as vertices enter or leave the stack.

(b) Vertex v has been colored but not yet explored, because the neighbors of every explored vertex have the proper color.

(c) Just before setting $s \leftarrow v$ in step B6, set PARENT$(v) \leftarrow u$, where PARENT is a new utility field. And just before terminating unsuccessfully in that step, do the following: "Repeatedly output NAME(u) and set $u \leftarrow$ PARENT(u), until $u =$ PARENT(v); then output NAME(u) and NAME(v)."

73. K_{10}. (And *random_graph*$(10,100,0,1,1,0,0,0,0,0)$ is J_{10}.)

74. **badness** has out-degree 22; no other vertices have out-degree > 20.

75. Let the parameters $(n_1, n_2, n_3, n_4, p, w, o)$ be respectively (a) $(n,0,0,0,-1,0,0)$; (b) $(n,0,0,0,1,0,0)$; (c) $(n,0,0,0,1,1,0)$; (d) $(n,0,0,0,-1,0,1)$; (e) $(n,0,0,0,1,0,1)$; (f) $(n,0,0,0,1,1,1)$; (g) $(m,n,0,0,1,0,0)$; (h) $(m,n,0,0,1,2,0)$; (i) $(m,n,0,0,1,3,0)$; (j) $(m,n,0,0,-1,0,0)$; (k) $(m,n,0,0,1,3,1)$; (l) $(n,0,0,0,2,0,0)$; (m) $(2,-n,0,0,1,0,0)$.

76. Yes, for example from C_1 and C_2 in answer 75(c). (But no self-loops can occur when $p < 0$, because arcs $x \longrightarrow y = x + k\delta$ are generated for $k = 1, 2, \ldots$ until y is out of range or $y = x$.)

77. Suppose x and y are vertices with $d(x,y) > 2$. Thus $x \not\!\!-\, y$; and if v is any other vertex we must have either $v \not\!\!-\, x$ or $v \not\!\!-\, y$. These facts yield a path of length at most 3 in $\overline{G}$ between any two vertices u and v.

78. (a) The number of edges, $\binom{n}{2}/2$, must be an integer. The smallest examples are K_0, K_1, P_4, C_5, and ⩘⩗.

(b) If q is any odd number, we have $u - v$ if and only if $\varphi^q(u) \not\!\!-\, \varphi^q(v)$. Therefore φ^q cannot have two fixed points, nor can it contain a 2-cycle.

(c) Such a permutation of V also defines a permutation $\widehat{\varphi}$ of the edges of K_n, taking $\{u,v\} \mapsto \widehat{\varphi}(\{u,v\}) = \{\varphi(u), \varphi(v)\}$, and it's easy to see that the cycle lengths of $\widehat{\varphi}$ are all multiples of 4. If $\widehat{\varphi}$ has t cycles, we obtain 2^t self-complementary graphs by painting the edges of each cycle with alternating colors.

(d) In this case φ has a unique fixed point v, and $G' = G \setminus v$ is self-complementary. Suppose φ has r cycles in addition to (v); then $\widehat{\varphi}$ has r cycles involving the edges that touch vertex v, and there are 2^r ways to extend G' to a graph G.

[*References:* H. Sachs, *Publicationes Mathematicæ* **9** (Debrecen, 1962), 270–288; G. Ringel, *Archiv der Mathematik* **14** (1963), 354–358.]

79. Solution 1, by H. Sachs, with $\varphi = (1\,2\,\ldots\,4k)$: Let $u \mathrel{-\!\!-} v$ when $u > v > 0$ and $u + v \bmod 4 \leq 1$; also $0 \mathrel{-\!\!-} v$ when $v \bmod 2 = 0$.

Solution 2, with $\varphi = (a_1\,b_1\,c_1\,d_1)\ldots(a_k\,b_k\,c_k\,d_k)$, where $a_j = 4j-3$, $b_j = 4j-2$, $c_j = 4j-1$, and $d_j = 4j$: Let $0 \mathrel{-\!\!-} b_j \mathrel{-\!\!-} a_j \mathrel{-\!\!-} c_j \mathrel{-\!\!-} d_j \mathrel{-\!\!-} 0$ for $1 \leq j \leq k$, and $a_i \mathrel{-\!\!-} a_j \mathrel{-\!\!-} b_i \mathrel{-\!\!-} d_j \mathrel{-\!\!-} c_i \mathrel{-\!\!-} c_j \mathrel{-\!\!-} d_i \mathrel{-\!\!-} b_j \mathrel{-\!\!-} a_i$, for $1 \leq i < j \leq k$.

80. (Solution by G. Ringel.) Let φ be as in answer 79, solution 2. Let E_0 be the $3k$ edges $b_j \mathrel{-\!\!-} a_j \mathrel{-\!\!-} c_j \mathrel{-\!\!-} d_j$ for $1 \leq j \leq k$; let E_1 be the $8\binom{k}{2}$ edges between $\{a_i, b_i, c_i, d_i\}$ and $\{b_j, d_j\}$ for $1 \leq i < j \leq k$; let E_2 be the $8\binom{k}{2}$ edges between $\{a_i, b_i, c_i, d_i\}$ and $\{a_j, c_j\}$ for $1 \leq i < j \leq k$. In case (a), $E_0 \cup E_1$ gives diameter 2; $E_0 \cup E_2$ gives diameter 3. Case (b) is similar, but we add $2k$ edges $b_j \mathrel{-\!\!-} 0 \mathrel{-\!\!-} d_j$ to E_1, $a_j \mathrel{-\!\!-} 0 \mathrel{-\!\!-} c_j$ to E_2.

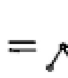
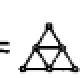
81. $\vec{C_3}$, $\vec{K_3}$, $D =$ , and $D^T =$ . (The *converse* D^T of a digraph D is obtained by reversing the direction of its arcs. There are 16 nonisomorphic simple digraphs of order 3 without loops, 10 of which are self-converse, including $\vec{C_3}$ and $\vec{K_3}$.)

82. (a) True, by definition. (b) True: If every vertex has d neighbors, every edge $u \mathrel{-\!\!-} v$ has $d-1$ neighbors $u \mathrel{-\!\!-} w$ and $d-1$ neighbors $w \mathrel{-\!\!-} v$. (c) True: $\{a_i, b_j\}$ has $m + n - 2$ neighbors, for $0 \leq i < m$ and $0 \leq j < n$. (d) False: $L(K_{1,1,2})$ has 5 vertices and 8 edges. (e) True. (f) True: The only nonadjacent edges are $\{0,1\} \not\mathrel{-\!\!-} \{2,3\}$, $\{0,2\} \not\mathrel{-\!\!-} \{1,3\}$, $\{0,3\} \not\mathrel{-\!\!-} \{1,2\}$. (g) True, for all $n > 0$. (h) False, unless G has no isolated vertices.

83. It is the Petersen graph. [A. Kowalewski, *Sitzungsberichte der Akademie der Wissenschaften in Wien*, Mathematisch-Nat. Klasse, Abteilung IIa, **126** (1917), 67–90.]

84. Yes: Let $\varphi(\{a_u, b_v\}) = \{a_{(u+v) \bmod 3}, b_{(u-v) \bmod 3}\}$ for $0 \leq u, v < 3$.

85. Let the vertex degrees be $\{d_1, \ldots, d_n\}$. Then G has $\frac{1}{2}(d_1 + \cdots + d_n)$ edges, and $L(G)$ has $\frac{1}{2}(d_1(d_1-1) + \cdots + d_n(d_n-1))$. Thus G and $L(G)$ both have exactly n edges if and only if $(d_1-2)^2 + \cdots + (d_n-2)^2 = 0$. Consequently exercise 58 gives the answer. [See V. V. Menon, *Canadian Math. Bull.* **8** (1965), 7–15.]

86. If $G =$  then $\overline{G} =$  $= L(G)$.

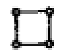
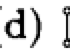
87. (a) Yes, easily. [In fact, R. L. Brooks has proved that *every* connected graph with maximum vertex degree $d > 2$ is d-colorable, except for the complete graph K_{d+1}; see *Proc. Cambridge Phil. Soc.* **37** (1941), 194–197.]

(b) No. There's essentially only one way to 3-color the edges of the outer 5-cycle in Fig. 2(e); this forces a conflict on the inner 5-cycle. [Petersen proved this in 1898.]

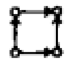
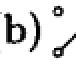
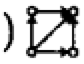
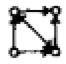
88. One cycle doesn't use the center vertex, and there are $(n-1)(n-2)$ cycles that do (namely, one for every ordered pair of distinct vertices on the rim). We don't count C_0.

89. Both sides equal $\begin{pmatrix} A & O & O \\ O & B & O \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ J & B & J \\ J & J & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ O & B & J \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & O & O \\ J & B & O \\ J & J & C \end{pmatrix}$, respectively.

90. K_4 and $\overline{K_4}$; $K_{1,1,2}$ and $\overline{K_{1,1,2}}$; $K_{2,2} = C_4$ and $\overline{K_{2,2}}$; $K_{1,3}$ and $\overline{K_{1,3}}$; $K_1 \oplus K_{1,2}$ and its complement; all graphs K_α are cographs by (47). Missing is $P_4 = \overline{P_4}$. (All connected subgraphs of a cograph have diameter ≤ 2; W_5 is a cograph, but not W_6.)

91. (a) ; (b) ; (c) ; (d) ; (e) ; (f) ; (g) . (In general we have $K_2 \triangle H = (K_2 \square H) \cup (K_2 \otimes \overline{H})$, and $K_2 \circ H = H \text{---} H$. Thus the coincidences $K_2 \triangle H = K_2 \square H$ and $K_2 \circ H = K_2 \boxtimes H$ occur if and only if H is a complete graph.)

Mnemonics: Our notations $G \square H$ and $G \boxtimes H$ nicely match diagrams (a) and (c), as suggested by J. Nešetřil, *Lecture Notes in Comp. Sci.* **118** (1981), 94–102. His analogous recommendation to write $G \times H$ for (b) is also tempting; but it wasn't adopted here, because hundreds of authors have used $G \times H$ to denote $G \square H$.

92. (a) ; (b) ; (c) ; (d) ; (e) .

93. $K_m \boxtimes K_n = K_m \circ K_n \cong K_{mn}$.

94. No; they're induced subgraphs of $K_{26} \square K_{26} \square K_{26} \square K_{26} \square K_{26}$.

95. (a) $d_u + d_v$. (b) $d_u d_v$. (c) $d_u d_v + d_u + d_v$. (d) $d_u(n - d_v) + (m - d_u)d_v$. (e) $d_u n + d_v$.

96. (a) $A \square B = A \otimes I + I \otimes B$. (b) $A \boxtimes B = A \square B + A \otimes B$. (c) $A \triangle B = A \otimes J + J \otimes B - 2A \otimes B$. (d) $A \circ B = A \otimes J + I \otimes B$. (Formulas (a), (b), and (d) define graph products of arbitrary digraphs and multigraphs. Formula (c) is valid in general for simple digraphs; but negative entries can occur when A and B contain values > 1.)

Historical notes: The direct product of matrices is often called the Kronecker product, because K. Hensel [*Crelle* **105** (1889), 329–344] said he had heard it in Kronecker's lectures; however, Kronecker never actually published anything about it. Its first known appearance was in a paper by J. G. Zehfuss [*Zeitschrift für Math. und Physik* **3** (1858), 298–301], who proved that $\det(A \otimes B) = (\det A)^n (\det B)^m$ when $m = m'$ and $n = n'$. The basic formulas $(A \otimes B)^T = A^T \otimes B^T$, $(A \otimes B)(A' \otimes B') = AA' \otimes BB'$, and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ are due to A. Hurwitz [*Math. Annalen* **45** (1894), 381–404].

97. Operations on adjacency matrices prove that $(G \oplus G') \square H = (G \square H) \oplus (G' \square H)$; $(G \oplus G') \boxtimes H = (G \boxtimes H) \oplus (G' \boxtimes H)$; $(G \oplus G') \circ H = (G \circ H) \oplus (G' \circ H)$. Since $G \square H \cong H \square G$, $G \otimes H \cong H \otimes G$, and $G \boxtimes H \cong H \boxtimes G$, we also have right-distributive laws $G \square (H \oplus H') \cong (G \square H) \oplus (G \square H')$; $G \otimes (H \oplus H') \cong (G \otimes H) \oplus (G \otimes H')$; $G \boxtimes (H \oplus H') \cong (G \boxtimes H) \oplus (G \boxtimes H')$. The lexicographic product satisfies $\overline{G \circ H} = \overline{G} \circ \overline{H}$; also $K_m \circ H = H \text{---} \cdots \text{---} H$, hence $K_m \circ \overline{K_n} = K_{n,\ldots,n}$. Furthermore $G \circ K_n = G \boxtimes K_n$; $K_m \otimes K_n = \overline{K_m \square K_n} = L(K_{m,n})$.

98. There are kl components (because of the distributive laws in the previous exercise, and the facts that $G \square H$ and $G \boxtimes H$ are connected when G and H are connected).

99. Every path from (u, v) to (u', v') in $G \square H$ must use at least $d_G(u, u')$ "G-steps" and at least $d_H(v, v')$ "H-steps"; and that minimum is achievable. Similar reasoning shows that $d_{G \boxtimes H}((u, v), (u', v')) = \max(d_G(u, u'), d_H(v, v'))$.

100. If G and H are connected, and if each of them has at least two vertices, $G \otimes H$ is disconnected if and only if G and H are bipartite. The "if" part is easy; conversely, if there's an odd cycle in G, we can get from (u, v) to (u', v') as follows: First go to (u'', v'), where u'' is any vertex of G that happens to be expedient. Then walk an even number of steps in G from u'' to u', while alternating in H between v' and one of its neighbors. [P. M. Weichsel, *Proc. Amer. Math. Soc.* **13** (1962), 47–52.]

101. Choose vertices u and v with maximum degree. Then $d_u + d_v = d_u d_v$ by exercise 95; so either $G = H = K_1$, or $d_u = d_v = 2$. In the latter case, $G = P_m$ or C_m, and $H = P_n$ or C_n. But $G \square H$ is connected, so G or H must be nonbipartite, say G. Then $G \square H$ is nonbipartite, so H must also be nonbipartite; thus $G = C_m$ and $H = C_n$, with m and n both odd. The shortest cycle in $C_m \square C_n$ has length $\min(m, n)$; in $C_m \otimes C_n$ it has length $\max(m, n)$; hence $m = n$. Conversely, if $n \geq 3$

is odd, we have $C_n \,\square\, C_n \cong C_n \otimes C_n$, under the isomorphism that takes $(u,v) \mapsto ((u+v) \bmod n, (u-v) \bmod n)$ for $0 \le u, v < n$. [D. J. Miller, *Canadian J. Math.* **20** (1968), 1511–1521.]

102. $P_m \boxtimes P_n$. (It is planar only when $\min(m,n) \le 2$ or $m = n = 3$.)

103.

1	2	3	4	5	7
2	1	3	4	6	8
3	1	2	5	6	8
4	1	2	5	6	
5	3	4	1	7	
6	2	3	4		
7	5	1			
8	2	3			

1	2	3	4	5	6	7	8	9
2	1	3	4	6	8	9		
3	1	2	5	6	8	9		
4	1	2	5	7				
5	3	4	1	7				
6	2	3	1	7				
7	4	5	6	1				
8	2	3	1	9				
9	8	2	3	1				

104. Edges must be created in a somewhat circuitous order, to maintain the tableau shape. Variables r and i mark the starting and ending row in column t. For example, the second part of exercise 103 begins with $i \leftarrow 1$, $t \leftarrow 8$, $r \leftarrow 1$; then $9 \,\text{—}\, 1$, $i \leftarrow 2$, $t \leftarrow 6$, $r \leftarrow 3$; then $9 \,\text{—}\, 3$, $9 \,\text{—}\, 2$, $i \leftarrow 4$, $t \leftarrow 4$, $r \leftarrow 8$; then $9 \,\text{—}\, 8$.

105. Notice that $d_k \ge k$ if and only if $c_k \ge k$. When $d_k \ge k$ we have

$$c_1 + \cdots + c_k \;=\; k^2 + \min(k, d_{k+1}) + \min(k, d_{k+2}) + \cdots + \min(k, d_n);$$

therefore the condition $d_1 + \cdots + d_k \le c_1 + \cdots + c_k - k$ is equivalent to

$$d_1 + \cdots + d_k \le f(k), \quad \text{where } f(k) = k(k-1) + \min(k, d_{k+1}) + \cdots + \min(k, d_n). \quad (*)$$

If $k \ge s$ we have $f(k+1) - f(k) = 2k - d_{k+1} \ge d_{k+1}$; hence $(*)$ holds for $1 \le k \le n$ if and only if it holds for $1 \le k \le s$. Condition $(*)$ was discovered by P. Erdős and T. Gallai [*Matematikai Lapok* **11** (1960), 264–274]. It is obviously necessary, if we consider the edges between $\{1, \ldots, k\}$ and $\{k+1, \ldots, n\}$.

Let $a_k = d_1 + \cdots + d_k - c_1 - \cdots - c_k + k$, and suppose that $a_k > 0$ for some $k \le s$ after steps H3 and H4 have acted. Let A_j, C_j, D_j, N, and S be the numbers that correspond to a_j, c_j, d_j, n, and s *before* steps H3 and H4; thus $N = n + 1$, $D_j = d_j + (0 \text{ or } 1)$, etc. We want to prove that $A_K > 0$ for some $K \le S$.

Steps H3 and H4 have removed the bottommost q cells in column t, for some $t \ge S$, together with the rightmost cells in rows 1 through p, where $q + p = D_N$. Thus $A_j = a_j$ for $1 \le j \le p$; furthermore $A_j = a_j$ when $j \ge C_t$.

Let k be minimal with $a_k > 0$, and let $d_k = d$; notice that $c_k \le d$. If $d > t$ we have $k \le p$, hence $A_k = a_k > 0$. Therefore we may assume that $d = t - (0 \text{ or } 1)$, and $D_k = t$.

If $k < j \le C_t$ we have $d_j \ge D_j - 1 = t - 1 \ge d - 1 \ge c_k - 1 \ge c_j - 1$. Therefore $A_K = a_K \ge a_k$ when $K = C_t$; we may assume that $C_t > S$.

Now $D_S = D_{S+1} = t$, so $S = t$. Also $k = t$; otherwise $c_k \ge S + 1 > t \ge d$. Therefore $s = S$ and $d = t = c_t$. Further analysis shows that the only possibility with $A_t \le 0$ is $D_j = t + [j \ge t]$ for $1 \le j \le N = t + 2$. Algorithm H does indeed change this "good" sequence into a "bad" one; but $D_1 + \cdots + D_N = t^2 + 3t - 1$ is odd.

106. False in the trivial cases when $d \le 1$ and $n \ge d + 2$. Otherwise true: In fact, the first $n - 1$ edges generated in step H4 contain no cycles, so they form a spanning tree.

107. The permutation φ of exercise 78 takes a vertex of degree d into a vertex of degree $n-1-d$. And φ^2 is an automorphism that pairs up two vertices of equal degree, except for a possible fixed point of degree $(n-1)/2$.

(Conversely, a somewhat intricate extension of Algorithm H will construct a self-complementary graph from every graphical sequence that satisfies these conditions,

provided that $d_{(n-1)/2} = (n-1)/2$ when n is odd. See C. R. J. Clapham and D. J. Kleitman, *J. Combinatorial Theory* **B20** (1976), 67–74.)

108. We may assume that $d_1^+ \geq \cdots \geq d_n^+$; the in-degrees d_k^- need not be in any particular order. Apply Algorithm H to the sequence $d_1 \ldots d_n = d_1^+ \ldots d_n^+$, but with the following changes: Step H2 becomes "[Done?] Terminate successfully if $d_1 = n = 0$; terminate unsuccessfully if $d_1 > n$." After setting i, t, and r in step H3, terminate unsuccessfully if $d_n^- > c_1$; otherwise do step H4 for $1 \leq j \leq d_n^-$, then set $n \leftarrow n-1$ and return to H2. In step H5, omit "$c_j \leftarrow c_j - 1$," and create the arc $k \longrightarrow n$ instead of the edge $k — n$. An argument like Lemma M and Corollary H justifies this approach.

(Exercise 7.2.1.4–57 proves that such digraphs exist if and only if $d_1^- + \cdots + d_n^- = d_1^+ + \cdots + d_n^+$ and $d_1^- \ldots d_n^- = \{d_1', \ldots, d_n'\}$, where $d_1' \geq \cdots \geq d_n'$ and $d_1' \ldots d_n'$ is majorized by the conjugate partition $c_1 \ldots c_n = (d_1^+ \ldots d_n^+)^T$. The variant where loops $v \longrightarrow v$ are forbidden is harder; see D. R. Fulkerson, *Pacific J. Math.* **10** (1960), 831–836.)

109. It's the same as exercise 108, if we put $d_k^+ = d_k[k \leq m]$ and $d_k^- = d_k[k > m]$.

110. There are p vertices of degree $d = d_1$ and q vertices of degree $d-1$, where $p+q = n$.

Case 1, $d = 2k+1$. Make $u — v$ whenever $(u-v) \bmod n \in \{2, 3, \ldots, k+1, n-k-1, \ldots, n-3, n-2\}$; also add the $p/2$ edges $1 — 2$, $3 — 4$, $\ldots$, $(p-1) — p$.

Case 2, $d = 2k$. Make $u — v$ whenever $(u-v) \bmod n \in \{2, 3, \ldots, k, n-k, \ldots, n-3, n-2\}$; also add the edges $1 — 2$, $\ldots$, $(q-1) — q$, as well as the path or cycle $(q = 0?\ n\colon q) — (q+1) — \cdots — (n-1) — n$. [D. L. Wang and D. J. Kleitman, in *Networks* **3** (1973), 225–239, have proved that such graphs are highly connected.]

111. Suppose $N = n + n'$ and $V' = \{n+1, \ldots, N\}$. We want to construct $e_k = d - d_k$ edges between k and V', and additional edges within V', so that each vertex of V' has degree d. Let $s = e_1 + \cdots + e_n$. This task is possible only if (i) $n' \geq \max(e_1, \ldots, e_n)$; (ii) $n'd \geq s$; (iii) $n'd \leq s + n'(n'-1)$; and (iv) $(n+n')d$ is even.

Such edges do exist whenever n' satisfies (i)–(iv): First, s suitable edges between V and V' can be created by cyclically choosing endpoints $(n+1, n+2, \ldots, n+n', n+1, \ldots)$, because of (i). This process assigns either $\lfloor s/n' \rfloor$ or $\lceil s/n' \rceil$ edges to each vertex of V'; we have $\lceil s/n' \rceil \leq d$ by (ii), and $d - \lfloor s/n' \rfloor \leq n' - 1$ by (iii). Therefore the additional edges needed inside V' are constructible by exercise 110 and (iv).

The choice $n' = n$ always works. Conversely, if $G = K_n(V) \setminus \{1 — 2\}$, condition (iii) requires $n' \geq n$ when $n \geq 4$. [P. Erdös and P. Kelly, *AMM* **70** (1963), 1074–1075.]

112. The uniquely best triangle in the *miles* data is

`Saint Louis, MO` $\overset{748}{—}$ `Toronto, ON` $\overset{746}{—}$ `Winston-Salem, NC` $\overset{748}{—}$ `Saint Louis, MO`.

113. By Murphy's Law, it has n rows and m columns; so it's $n \times m$, not $m \times n$.

114. A loop in a multigraph is an edge $\{a, a\}$ with repeated vertices, and a multigraph is a 2-uniform hypergraph. Thus we should allow the incidence matrix of a general hypergraph to have entries greater than 1 when an edge contains a vertex more than once. (A pedant would probably call this a "multihypergraph.") With these considerations in mind, the incidence matrix and bipartite graph corresponding to (26) are

$$\begin{pmatrix} 210000 \\ 011100 \\ 001122 \end{pmatrix};$$



115. The element in row e and column f of B^TB is $\sum_v b_{ve}b_{vf}$; so B^TB is $2I$ plus the adjacency matrix of $L(G)$. Similarly, BB^T is D plus the adjacency matrix of G, where D is the diagonal matrix with d_{vv} = degree of v. (See exercises 2.3.4.2–18, 19, and 20.)

116. $\overline{K_{m,n}^{(r)}} = K_m^{(r)} \oplus K_n^{(r)}$, generalizing (38), for all $r \geq 1$.

117. The nonisomorphic multisets of singleton edges for $m = 4$ and $V = \{0,1,2\}$ are $\{\{0\},\{0\},\{0\},\{0\}\}$, $\{\{0\},\{0\},\{0\},\{1\}\}$, $\{\{0\},\{0\},\{1\},\{1\}\}$, and $\{\{0\},\{0\},\{1\},\{2\}\}$. The answer in general is the number of partitions of m into at most n parts, namely $\left|{m+n \atop n}\right|$, using the notation explained in Section 7.2.1.4. (Of course, there's little reason to think of partitions as 1-uniform hypergraphs, except when answering strange exercises.)

118. Let d be the sum of the vertex degrees. The corresponding bipartite graph is a forest with $m+n$ vertices, d edges, and p components. Hence $d = m + n - p$, by Theorem 2.3.4.1A.

119. Then there's an additional edge, containing all seven vertices.

120. We could say that (hyper)arcs are arbitrary sequences of vertices, or sequences of distinct vertices. But most authors seem to define hyperarcs to be $A \longrightarrow v$, where A is an unordered set of vertices. When the best definition is found, it will probably be the one that has the most important practical applications.

121. $\chi(H) = |F| - \alpha(I(H)^T)$ is the size of a minimum cover of V by sets of F.

122. (a) One can verify that there are just seven 3-element covers, namely the vertices of an edge; so there are seven 4-element independent sets, namely the complements of an edge. We can't two-color the hypergraph, because one color would need to be used 4 times and the other three colors would be an edge. (Hypergraph (56) is essentially the projective plane with seven points and seven lines.)

(b) Since we're dualizing, let's call the vertices and edges of the Petersen graph "points" and "lines"; then the vertices and edges of the dual are lines and points, respectively. Color red the five lines that join an outer point to an inner point. The other ten lines are independent (they don't contain all three of the lines touching any point); so they can be colored green. No set of eleven lines can be independent, because no four lines can touch all ten points. (Thus the Petersen dual is a bipartite hypergraph, in spite of the fact that it contains cycles of length 5.)

123. They correspond to $n \times n$ latin squares, whose entries are the vertex colors.

124. Four colors easily suffice. If it were 3-colorable, there must be four vertices of each color, since no five vertices are independent. Then two opposite corners must have the same color, and a contradiction arises quickly.

125. The Chvátal graph is the smallest such graph with $g = 4$. G. Brinkmann found the smallest with $g = 5$: It has 21 vertices a_j, b_j, c_j for $0 \leq j < 7$, with edges $a_j \longrightarrow a_{j+2}$, $a_j \longrightarrow b_j$, $a_j \longrightarrow b_{j+1}$, $b_j \longrightarrow c_j$, $b_j \longrightarrow c_{j+2}$, $c_j \longrightarrow c_{j+3}$ and subscripts mod 7. M. Meringer showed that there must be at least 35 vertices if $g > 5$. B. Grünbaum conjectured that g can be arbitrarily large; but no further constructions are known. [See *AMM* **77** (1970), 1088–1092; *Graph Theory Notes of New York* **32** (1997), 40–41.]

126. When m and n are even, both C_m and C_n are bipartite, and 4-coloring is easy. Otherwise a 4-coloring is impossible. When $m = n = 3$, a 9-coloring is optimum by exercise 93. When $m = 3$ and $n = 4$ or 5, at most two vertices are independent; it's easy to find an optimum 6- or 8-coloring. Otherwise we obtain a 5-coloring by painting vertex (j,k) with $(a_j + 2b_k) \bmod 5$, where periodic sequences $\langle a_j \rangle$ and $\langle b_k \rangle$ exist with period lengths m and n, respectively, such that $a_j - a_{j+1} \equiv \pm 1$ and $b_k - b_{k+1} \equiv \pm 1$ for all j and k. [K. Vesztergombi, *Acta Cybernetica* **4** (1978), 207–212.]

127. (a) The result is true when $n = 1$. Otherwise let $H = G \setminus v$, where v is any vertex. Then $\overline{H} = \overline{G} \setminus v$, and we have $\chi(H) + \chi(\overline{H}) \leq n$ by induction. Clearly $\chi(G) \leq \chi(H) + 1$

and $\chi(\overline{G}) \le \chi(\overline{H}) + 1$; so there's no problem unless equality holds in all three cases. But that can't happen; it implies that $\chi(H) \le d$ and $\chi(\overline{H}) \le n-1-d$, where d is the degree of v in G. [E. A. Nordhaus and J. W. Gaddum, *AMM* **63** (1956), 175–177.]

To get equality, let $G = K_a \oplus \overline{K_b}$, where $ab > 0$ and $a+b=n$. Then we have $\overline{G} = \overline{K_a} — K_b$, $\chi(G) = a$, and $\chi(\overline{G}) = b+1$. [All graphs for which equality holds have been found by H.-J. Finck, *Wiss. Zeit. der Tech. Hochschule Ilmenau* **12** (1966), 243–246.]

(b) A k-coloring of G has at least $\lceil n/k \rceil$ vertices of some color; those vertices form a clique in $\overline{G}$. Hence $\chi(G)\chi(\overline{G}) \ge \chi(G)\lceil n/\chi(G)\rceil \ge n$. Equality holds when $G = K_n$.

(From (a) and (b) we deduce that $\chi(G)+\chi(\overline{G}) \ge 2\sqrt{n}$ and $\chi(G)\chi(\overline{G}) \le \frac{1}{4}(n+1)^2$.)

128. $\chi(G \,\square\, H) = \max(\chi(G), \chi(H))$. This many colors is clearly necessary. And if the functions $a(u)$ and $b(v)$ color G and H with the colors $\{0,1,\ldots,k-1\}$, we can color $G \,\square\, H$ with $c(u,v) = (a(u)+b(v)) \bmod k$.

129. A complete row or column (16 cases); a complete diagonal of length 4 or more (18 cases); a 5-cell pattern $\{(x,y),(x-a,y-a),(x-a,y+a),(x+a,y-a),(x+a,y+a)\}$ for $a \in \{1,2,3\}$ $(36+16+4$ cases); a 5-cell pattern $\{(x,y),(x-a,y),(x+a,y),(x,y-a),$ $(x,y+a)\}$ for $a \in \{1,2,3\}$ $(36+16+4$ cases); a pattern containing four of those five cells, when the fifth lies off the board $(24+32+24$ cases); or a 4-cell pattern $\{(x,y),(x+a,y),(x,y+a),(x+a,y+a)\}$ for $a \in \{1,3,5,7\}$ $(49+25+9+1$ cases). Altogether 310 maximal cliques, with respectively $(168,116,4,4,18)$ of size $(4,5,6,7,8)$.

130. If graph G has p maximal cliques and graph H has q, then the join $G — H$ has pq, because the cliques of $G — H$ are simply the unions of cliques from G and H. Furthermore, the empty graph $\overline{K_n}$ has n maximal cliques (namely its singleton sets).

Thus the complete k-partite graph with part sizes $\{n_1,\ldots,n_k\}$, being the join of empty graphs of those sizes, has $n_1 \ldots n_k$ maximal cliques.

131. Assume that $n > 1$. In a complete k-partite graph, the number $n_1 \ldots n_k$ is maximized when each part has size 3, except perhaps for one or two parts of size 2. (See exercise 7.2.1.4–68(a).) So we must prove that $N(n)$ cannot be larger than this in *any* graph.

Let $m(v)$ be the number of maximal cliques that contain vertex v. If $u \not— v$ and $m(u) \le m(v)$, construct the graph G' that is like G except that u is now adjacent to all the neighbors of v instead of to its former neighbors. Every maximal clique U in either graph belongs to one of three classes:

i) $u \in U$; there are $m(u)$ of these in G and $m(v)$ of them in G'.

ii) $v \in U$; there are $m(v)$ of these in G and also in G'.

iii) $u \notin U$ and $v \notin U$; such maximal cliques in G are also maximal in G'.

Therefore G' has at least as many maximal cliques as G. And we can obtain a complete k-partite graph by appropriately repeating the process.

[This argument, due to Paul Erdős, was presented by J. W. Moon and L. Moser in *Israel J. Math.* **3** (1965), 23–25.]

132. The strong product of cliques in G and H is a clique in $G \boxtimes H$, by exercise 93; hence $\omega(G \boxtimes H) \ge \omega(G)\omega(H) = \chi(G)\chi(H)$. On the other hand, colorings $a(u)$ and $b(v)$ of G and H lead to the coloring $c(u,v) = (a(u), b(v))$ of $G \boxtimes H$; hence $\chi(G \boxtimes H) \le \chi(G)\chi(H)$. And $\omega(G \boxtimes H) \le \chi(G \boxtimes H)$.

133. (a) 24; (b) 60; (c) 3; (d) 6; (e) 6; (f) 4; (g) 5; (h) 4; (i) $K_2 \boxtimes C_{12}$; (j) 18; (k) 12. (l) Yes, of degree 5. (m) No. [Can it be drawn with fewer than 12 crossings?] (n) Yes; in fact, it is 4-connected (see Section 7.4.1). (o) Yes; we consider *every* graph to be directed, with two arcs for each edge. (p) Of course not. (q) Yes, easily.

[The musical graph represents simple modulations between key signatures. It appears on page 73 of *Graphs* by R. J. Wilson and J. J. Watkins (1990).]

134. By rotating and/or swapping the inner and outer vertices, we can find an automorphism that takes any vertex into C. If C is fixed, we can interchange the inner and outer vertices of any subset of the remaining 11 pairs, and/or do a left-right reflection. Therefore there are $24 \times 2^{11} \times 2 = 98{,}304$ automorphisms altogether.

135. Let $\omega = e^{2\pi i/6}$, and define the matrices $Q = (q_{ij})$, $S = (s_{ij})$, where $q_{ij} = [j = (i+1) \bmod 12]$ and $s_{ij} = \omega^{ij}$, for $0 \le i, j < 12$. By exercise 96(b), the adjacency matrix of the musical graph $K_2 \boxtimes C_{12}$ is $A = \binom{1\,1}{1\,1} \otimes (I+Q+Q^-) - I$. Let T be the matrix $\binom{1\;\;\,1}{1\;-1} \otimes S$; then $T^- AT$ is a diagonal matrix D whose first 12 entries are $1 + 4\cos\frac{j\pi}{6}$ for $0 \le j < 12$, and whose other 12 entries are -1. Therefore $A^{2m} = TD^mT^-$, and it follows that the number of $2m$-step walks from C to (C, G, D, A, E, B, F♯) respectively is

$$\begin{aligned}
\mathrm{C}_m &= \tfrac{1}{24}(25^m + 2(13+4\sqrt{3})^m + 3^{2m+1} + 2(13-4\sqrt{3})^m + 16);\\
\mathrm{G}_m &= \tfrac{1}{24}(25^m + \sqrt{3}(13+4\sqrt{3})^m - \sqrt{3}(13-4\sqrt{3})^m - 1);\\
\mathrm{D}_m &= \tfrac{1}{24}(25^m + (13+4\sqrt{3})^m + (13-4\sqrt{3})^m - 3);\\
\mathrm{A}_m &= \tfrac{1}{24}(25^m - 3^{2m+1} + 2);\\
\mathrm{E}_m &= \tfrac{1}{24}(25^m - (13+4\sqrt{3})^m - (13-4\sqrt{3})^m + 1);\\
\mathrm{B}_m &= \tfrac{1}{24}(25^m - \sqrt{3}(13+4\sqrt{3})^m + \sqrt{3}(13-4\sqrt{3})^m - 1);\\
\mathrm{F}^\sharp_m &= \tfrac{1}{24}(25^m - 2(13+4\sqrt{3})^m + 3^{2m+1} - 2(13-4\sqrt{3})^m);
\end{aligned}$$

also $\mathrm{a}_m = \mathrm{C}_m - 1$, $\mathrm{d}_m = \mathrm{F}_m = \mathrm{e}_m = \mathrm{G}_m$, etc. In particular, $(\mathrm{C}_6, \mathrm{G}_6, \mathrm{D}_6, \mathrm{A}_6, \mathrm{E}_6, \mathrm{B}_6, \mathrm{F}^\sharp_6) =$ (15462617, 14689116, 12784356, 10106096, 7560696, 5655936, 5015296), so the desired probability is $15462617/5^{12} \approx 6.33\%$. As $m \to \infty$, the probabilities are all $\frac{1}{24} + O(0.8^m)$.

136. No. Only two Cayley graphs of order 10 are cubic, namely $K_2 \square C_5$ (whose vertices can be written $\{e, \alpha, \alpha^2, \alpha^3, \alpha^4, \beta, \beta\alpha, \beta\alpha^2, \beta\alpha^3, \beta\alpha^4\}$ where $\alpha^5 = \beta^2 = (\alpha\beta)^2 = e$) and the graph with vertices $\{0, 1, \ldots, 9\}$ and arcs $v \to (v \pm 1) \bmod 10$, $v \to (v+5) \bmod 10$. [See D. A. Holton and J. Sheehan, *The Petersen Graph* (1993), exercise 9.10. Incidentally, the SGB graphs $\mathit{raman}(p, q, t, 0)$ are Cayley graphs.]

137. Let $[x, y]$ denote the label of (x, y); we want $[x, y] = [x+a, y+b] = [x+c, y+d]$ for all x and y. If A is the matrix $\binom{a\;b}{c\;d}$, the operation of adding t times the bottom row of A to the top row changes A to the matrix $A' = \binom{1\;t}{0\;1}A = \binom{a'\;b'}{c'\;d'}$, where $a' = a + tc$, $b' = b + td$, $c' = c$, $d' = d$. The new condition $[x, y] = [x + a', y + b'] = [x + c', y + d']$ is equivalent to the old; and $\gcd(a', b', c', d') = \gcd(a, b, c, d)$. Similarly we can premultiply A by $\binom{1\;0}{t\;1}$ without really changing the problem.

We can also operate on columns, replacing A by $A'' = A\binom{1\;t}{0\;1} = \binom{a''\;b''}{c''\;d''}$, where $a'' = a$, $b'' = ta+b$, $c'' = c$, $d'' = tc+d$. This operation does alter the problem, but only slightly: If we find a labeling that satisfies $[\![x, y]\!] = [\![x + a'', y + b'']\!] = [\![x + c'', y + d'']\!]$ for all x and y, then we'll have $[x, y] = [x+a, y+b] = [x+c, x+d]$ if $[x, y] = [\![x, y+tx]\!]$. Similarly we can postmultiply A by $\binom{1\;0}{t\;1}$; the problem remains almost the same.

A series of such row and column operations will reduce A to the simple form $UAV = \binom{1\;0}{0\;n}$, where U and V are integer matrices with $\det U = \det V = 1$. And if we have $V = \binom{\alpha\;\beta}{\gamma\;\delta}$, a labeling for the reduced problem that satisfies the simple conditions $[\![x, y]\!] = [\![x+1, y]\!] = [\![x, y+n]\!]$ will provide a solution to the original labeling problem if we define $[x, y] = [\![\alpha x + \gamma y, \beta x + \delta y]\!]$.

Finally, the reduced labeling problem is easy: We let $[\![x,y]\!] = y \bmod n$. Thus the desired answer is to set $p = \beta$, $q = \delta$.

138. Proceeding as before, but with a $k \times k$ matrix A, row and column operations will reduce the problem to a diagonal matrix UAV. The diagonal entries $(d_1, \ldots, d_k)$ are characterized by the condition that $d_1 \ldots d_j$ is the greatest common divisor of the determinants of all $j \times j$ submatrices of A. [This is "Smith normal form"; see H. J. S. Smith, *Philosophical Transactions* **151** (1861), 293–326, §14.] If the labeling $[\![x]\!]$ satisfies the reduced problem, the original problem is satisfied by $[x] = [\![xV]\!]$. The number of elements in the generalized torus is $n = \det A = d_1 \ldots d_k$.

The reduced problem has a simple solution as before if $d_1 = \cdots = d_{k-1} = 1$. But in general the reduced labeling will be an r-dimensional ordinary torus of dimensions $(d_{k-r+1}, \ldots, d_k)$, where $d_{k-r+1} > d_{k-r} = 1$. (Here $d_0 = 1$; we might have $r = k$.)

In the requested example, we find $d_1 = 1$, $d_2 = 2$, $d_3 = 10$, $n = 20$; indeed,

$$UAV = \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & -1 \\ -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & 5 & 6 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 10 \end{pmatrix}.$$

Each point (x, y, z) now receives a two-dimensional label $(u, v) = ((5x + y) \bmod 2, (6x + y + z) \bmod 10)$. The six neighbors of (u, v) are $((u \pm 1) \bmod 2, v)$, $((u \pm 1) \bmod 2, (v \pm 1) \bmod 10)$, $(u, (v \pm 1) \bmod 10)$. It's a multigraph, since the first two neighbors are identical; but it's not the same as the multigraph $C_2 \boxtimes C_{10}$, which has degree 8.

[Generalized toruses are essentially the Cayley graphs of Abelian groups; see exercise 136. They have been proposed as convenient interconnection networks, in which case it is desirable to minimize the diameter when k and n are given. See C. K. Wong and D. Coppersmith, *JACM* **21** (1974), 392–402; C. M. Fiduccia, R. W. Forcade, and J. S. Zito, *SIAM J. Discrete Math.* **11** (1998), 157–167.]

139. (This exercise helps clarify the distinction between labeled graphs G, in which the vertices have definite names, and unlabeled graphs H such as those in Fig. 2.) If N_H is the number of labeled graphs on $\{1, 2, \ldots, h\}$ that are isomorphic to H, and if U is any h-element subset of V, the probability that $G \mid U$ is isomorphic to H is $N_H/2^{h(h-1)/2}$. Therefore the answer is $\binom{n}{h} N_H/2^{h(h-1)/2}$. We need only figure out the value of N_H, which is: (a) 1; (b) $h!/2$; (c) $(h-1)!/2$; (d) $h!/a$, where H has a automorphisms.

140. (a) $\#(K_3{:}W_n) = n-1$ and $\#(P_3{:}W_n) = \binom{n-1}{2}$ for $n \geq 5$; also $\#(\overline{K_3}{:}W_8) = 7$.

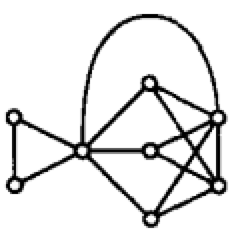
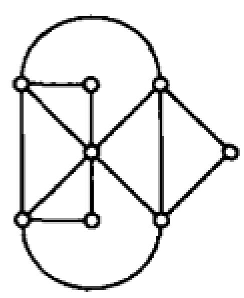
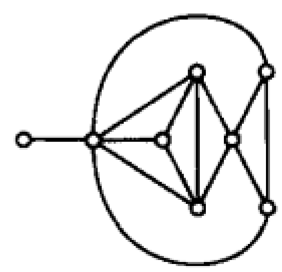
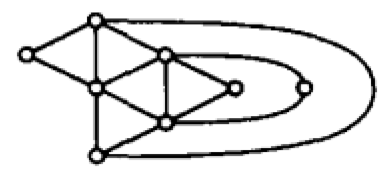
(b) G is proportional if and only if $\#(K_3{:}G) = \#(\overline{K_3}{:}G) = \frac{1}{8}\binom{n}{3}$ and $\#(P_3{:}G) = \#(\overline{P}_3{:}G) = \frac{3}{8}\binom{n}{3}$. If G has e edges, we have $(n-2)e = 3\#(K_3{:}G) + 2\#(P_3{:}G) + \#(\overline{P}_3{:}G)$, because every pair of vertices appears in $n-2$ induced subgraphs. If G has degree sequence $d_1 \ldots d_n$, we have $d_1 + \cdots + d_n = 2e$, $\binom{d_1}{2} + \cdots + \binom{d_n}{2} = 3\#(K_3{:}G) + \#(P_3{:}G)$, and $d_1(n-1-d_1) + \cdots + d_n(n-1-d_n) = 2\#(P_3{:}G) + 2\#(\overline{P}_3{:}G)$. Therefore a proportional graph satisfies $(*)$ — unless $n = 2$. (The exercise should have excluded that case.)

Conversely, if G satisfies $(*)$ and has the correct $\#(K_3{:}G)$, it also has the correct $\#(P_3{:}G)$, $\#(\overline{P}_3{:}G)$, and $\#(\overline{K_3}{:}G)$.

[*References:* S. Janson and J. Kratochvíl, *Random Structures & Algorithms* **2** (1991), 209–224. In *J. Combinatorial Theory* **B47** (1989), 125–145, A. D. Barbour, M. Karoński, and A. Ruciński had shown that the variance of $\#(H{:}G)$ is proportional to either n^{2h-2}, n^{2h-3}, or n^{2h-4}, where the first case occurs when H does not have $\frac{1}{2}\binom{h}{2}$ edges, and the third case occurs when H is a proportional graph.]

141. Only 8 degree sequences $d_1 \ldots d_8$ satisfy $(*)$: 73333333 (1/2), 65433322 (26/64), 64444222 (2/10), 64443331 (8/22), 55543222 (8/20), 55533331 (2/10), 55444321 (26/64),

and 44444440 (1/2). Each degree sequence is shown here with statistics (N_1/N), where N nonisomorphic graphs have that sequence and N_1 of them are proportional. The last three cases are complements of the first three. No graph of order 8 is both proportional and self-complementary. Maximally symmetric examples of the first five cases are W_8,

, , , and .

142. The hint follows as in the previous answer; $(n-3)\#(\overline{K}_3{:}G)$ and $(n-3)\#(P_3{:}G)$ can also be expressed in terms of four-vertex counts. Furthermore, a graph with e edges has $\binom{e}{2} = \#(P_3 \subseteq G) + \#(K_2 \oplus K_2 \subseteq G)$, because any two edges form either P_3 or $K_2 \oplus K_2$; in this formula, $\#(P_3 \subseteq G)$ counts not-necessarily-induced subgraphs.

We have $\#(P_3 \subseteq G) = \#(P_3{:}G) + 3\#(K_3{:}G)$, and a similar formula expresses $\#(K_2 \oplus K_2 \subseteq G)$ in terms of induced counts. Thus an extraproportional graph must be proportional and satisfy $e = \frac{1}{2}\binom{n}{2}$, $\#(P_3 \subseteq G) = \frac{3}{4}\binom{n}{3}$, $\#(K_2 \oplus K_2 \subseteq G) = \frac{3}{4}\binom{n}{4}$. But these values contradict the formula for $\binom{e}{2}$.

143. Consider the graph whose vertices are the rows of A, and whose edges u — v signify that rows u and v agree except in one column, j. Label such an edge j.

If the graph contains a cycle, delete any edge of the cycle, and repeat the process until no cycles remain. Notice that the label on every deleted edge appears elsewhere in its cycle; hence the deletions don't affect the set of edge labels. But we're left with fewer than $m \le n$ edges, by Theorem 2.3.4.1A; so there are fewer than n different labels. [See J. A. Bondy, *J. Combinatorial Theory* **B12** (1972), 201–202.]

144. Let G be the graph on vertices $\{1,\ldots,m\}$, with edges i — j if and only if $* \ne x_{il} \ne x_{jl} \ne *$ for some l. This graph is k-colorable if and only if there is a completion with at most k distinct rows. Conversely, if G is a graph on vertices $\{1,\ldots,n\}$, with adjacency matrix A, the $n \times n$ matrix $X = A + *(J - I - A)$ has the property that i — j if and only if $* \ne x_{il} \ne x_{jl} \ne *$ for some l. [See M. Sauerhoff and I. Wegener, *IEEE Trans.* **CAD-15** (1996), 1435–1437.]

145. Set $c \leftarrow 0$ and repeat the following operations for $1 \le j \le n$: If $c = 0$, set $x \leftarrow a_j$ and $c \leftarrow 1$; otherwise if $x = a_j$, set $c \leftarrow c + 1$; otherwise set $c \leftarrow c - 1$. Then x is the answer. The idea is to keep track of a possible majority element x, which occurs c times in nondiscarded elements; we discard a_j and one x whenever finding $x \ne a_j$. [See *Automated Reasoning* (Kluwer, 1991), 105–117. Extensions to find all elements that occur more than n/k times, in $O(n \log k)$ steps, have been discussed by J. Misra and D. Gries, *Science of Computer Programming* **2** (1982), 143–152.]

SECTION 7.1.1

1. (Solution by C. Sartena.) He was describing the implication $x \Rightarrow y$, with "it" standing respectively for y, x, x, y, y, x. (Other solutions are possible.)

2. The Earth operation corresponding to the Pincusian $x \circ y$ is $\overline{\bar{x} \circ \bar{y}}$; its truth table is therefore the reverse of the complement of the truth table for $\circ$. Hence the respective answers are $\top$, $\lor$, $\subset$, L, $\supset$, R, $\equiv$, $\land$, $\barwedge$, $\oplus$, $\bar{\mathrm{R}}$, $\bar\supset$, $\bar{\mathrm{L}}$, $\bar\subset$, $\bar\lor$, $\perp$. (Any identity involving the 16 operations of Table 1 implies a corresponding dual identity obtained by substituting the Pincusian equivalents. For example, each of De Morgan's laws (11) and (12) is the dual of the other, as are the identities (3), (4) relating $\equiv$ and $\oplus$. In this sense $\equiv$ can be considered to be just as useful as its dual, $\oplus$.)

3. (a) $\lor$; (b) $\land$; (c) $\bar{\mathrm{L}}$; (d) $\equiv$. [Many formulas actually work out better if we use -1 for truth and $+1$ for falsehood, even though this convention seems a bit immoral; then $x \cdot y$ corresponds to $\oplus$. Notice that $\langle xyz \rangle = \mathrm{sign}(x+y+z)$, with either convention.]

4. [*Trans. Amer. Math. Soc.* **14** (1913), 481–488.] (a) Start with the truth tables for L and R; then compute truth table $\alpha \barwedge \beta$ bitwise from each known pair of truth tables α and β, generating the results in order of the length of each formula and writing down a shortest formula that leads to each new 4-bit table:

$\perp$: $(x \barwedge (x \barwedge x)) \barwedge (x \barwedge (x \barwedge x))$	$\bar\lor$: $(x \barwedge (x \barwedge x)) \barwedge ((y \barwedge y) \barwedge (x \barwedge x))$
$\land$: $(x \barwedge y) \barwedge (x \barwedge y)$	$\equiv$: $(x \barwedge y) \barwedge ((y \barwedge y) \barwedge (x \barwedge x))$
$\bar\supset$: $(x \barwedge (x \barwedge y)) \barwedge (x \barwedge (x \barwedge y))$	$\bar{\mathrm{R}}$: $y \barwedge y$
L: x	$\subset$: $y \barwedge (x \barwedge x)$
$\bar\subset$: $(y \barwedge (x \barwedge x)) \barwedge (y \barwedge (x \barwedge x))$	$\bar{\mathrm{L}}$: $x \barwedge x$
R: y	$\supset$: $x \barwedge (x \barwedge y)$
$\oplus$: $(y \barwedge (x \barwedge x)) \barwedge (x \barwedge (x \barwedge y))$	$\barwedge$: $x \barwedge y$
$\lor$: $(y \barwedge y) \barwedge (x \barwedge x)$	$\top$: $x \barwedge (x \barwedge x)$

(b) In this case we start with four tables $\perp$, $\top$, L, R, and we prefer formulas with fewer occurrences of variables whenever there's a choice between formulas of a given length:

$\perp$: 0	$\bar\lor$: $1 \barwedge ((y \barwedge 1) \barwedge (x \barwedge 1))$
$\land$: $(x \barwedge y) \barwedge 1$	$\equiv$: $(x \barwedge y) \barwedge ((y \barwedge 1) \barwedge (x \barwedge 1))$
$\bar\supset$: $((y \barwedge 1) \barwedge x) \barwedge 1$	$\bar{\mathrm{R}}$: $y \barwedge 1$
L: x	$\subset$: $y \barwedge (x \barwedge 1)$
$\bar\subset$: $(y \barwedge (x \barwedge 1)) \barwedge 1$	$\bar{\mathrm{L}}$: $x \barwedge 1$
R: y	$\supset$: $(y \barwedge 1) \barwedge x$
$\oplus$: $(y \barwedge (x \barwedge 1)) \barwedge ((y \barwedge 1) \barwedge x)$	$\barwedge$: $x \barwedge y$
$\lor$: $(y \barwedge 1) \barwedge (x \barwedge 1)$	$\top$: 1

5. (a) $\perp$: $x \mathbin{\bar\subset} x$; $\land$: $(x \mathbin{\bar\subset} y) \mathbin{\bar\subset} y$; $\bar\supset$: $y \mathbin{\bar\subset} x$; L: x; $\bar\subset$: $x \mathbin{\bar\subset} y$; R: y; the other 10 cannot be expressed. (b) With constants, however, all 16 are possible:

$\perp$: 0	$\bar\lor$: $y \mathbin{\bar\subset} (x \mathbin{\bar\subset} 1)$
$\land$: $(y \mathbin{\bar\subset} 1) \mathbin{\bar\subset} x$	$\equiv$: $(y \mathbin{\bar\subset} x) \mathbin{\bar\subset} ((x \mathbin{\bar\subset} y) \mathbin{\bar\subset} 1)$
$\bar\supset$: $y \mathbin{\bar\subset} x$	$\bar{\mathrm{R}}$: $y \mathbin{\bar\subset} 1$
L: x	$\subset$: $(x \mathbin{\bar\subset} y) \mathbin{\bar\subset} 1$
$\bar\subset$: $x \mathbin{\bar\subset} y$	$\bar{\mathrm{L}}$: $x \mathbin{\bar\subset} 1$
R: y	$\supset$: $(y \mathbin{\bar\subset} x) \mathbin{\bar\subset} 1$
$\oplus$: $((y \mathbin{\bar\subset} x) \mathbin{\bar\subset} ((x \mathbin{\bar\subset} y) \mathbin{\bar\subset} 1)) \mathbin{\bar\subset} 1$	$\barwedge$: $((y \mathbin{\bar\subset} 1) \mathbin{\bar\subset} x) \mathbin{\bar\subset} 1$
$\lor$: $(y \mathbin{\bar\subset} (x \mathbin{\bar\subset} 1)) \mathbin{\bar\subset} 1$	$\top$: 1

[B. A. Bernstein, *University of California Publications in Mathematics* **1** (1914), 87–96.]

6. (a) $\bot$, $\land$, L, R, $\oplus$, $\lor$, $\equiv$, $\top$. (b) $\bot$, L, R, $\oplus$, $\equiv$, $\top$. [Notice that all of these operators are associative. In fact, the stated identity implies the associative law in general: First we have (i) $(x \circ y) \circ ((z \circ y) \circ w) = ((x \circ z) \circ (z \circ y)) \circ ((z \circ y) \circ w) = (x \circ z) \circ w$, and similarly (ii) $(x \circ (y \circ z)) \circ (y \circ w) = x \circ (z \circ w)$. Furthermore (iii) $(x \circ y) \circ (z \circ w) = (x \circ y) \circ ((z \circ y) \circ (y \circ w)) = (x \circ z) \circ (y \circ w)$ by (i). Thus $(x \circ z) \circ w = (x \circ z) \circ ((z \circ z) \circ w) = (x \circ (z \circ z)) \circ (z \circ w) = x \circ (z \circ w)$ by (i), (iii), (ii). The free system generated by $\{x_1, \ldots, x_n\}$ has exactly $n + 2^n n^2$ distinct elements, namely $\{x_j \mid 1 \le j \le n\}$ and $\{x_i \circ x_{j_1} \circ \cdots \circ x_{j_r} \circ x_k \mid r \ge 0 \text{ and } 1 \le i, k \le n \text{ and } 1 \le j_1 < \cdots < j_r \le n\}$.]

7. Equivalently, we want the identity $y \circ (x \circ y) = x$, which holds only for $\oplus$ and $\equiv$. [Jevons noticed this property of $\oplus$ in *Pure Logic* §151, but he did not pursue the matter. We will investigate general systems of this nature, called "gropes," in Section 7.2.3.]

8. $(\{\bot, \land, \bar{\subset}\}, S_0)$, $(\{\top, \lor, \supset\}, S_1)$, $(\{\mathsf{L}, \bar{\mathsf{L}}\}, S_0 \cap S_1)$, $(\{\oplus, \equiv, \bar{\mathsf{R}}\}, S_2)$, $(\{\bar{\supset}, \bar{\lor}\}, S_0 \cap S_2)$, $(\{\subset, \bar{\land}\}, S_1 \cap S_2)$, and (R, any), where $S_0 = \{\square \mid 0 \,\square\, 0 = 0\}$, $S_1 = \{\square \mid 1 \,\square\, 1 = 1\}$, and $S_2 = \{\square \mid \bar{x} \,\square\, \bar{y} = \overline{x \,\square\, y}\} = \{\mathsf{L}, \mathsf{R}, \bar{\mathsf{L}}, \bar{\mathsf{R}}\}$. Thus 92 of the 256 pairs are left-distributive. [This problem and those of exercise 6 were first treated by E. Schröder in §55 of his posthumously published *Vorlesungen über die Algebra der Logik* **2**, 2 (1905). He expressed the answer by saying in essence that the respective truth tables $(pqrs, wxyz)$ of $(\circ, \square)$ must satisfy the relation $((pq \lor rs) \land \bar{z}) \lor ((\bar{p}\bar{q} \lor \bar{r}\bar{s}) \land w) \lor ((p\bar{q} \lor r\bar{s}) \land ((w \equiv z) \lor (x \equiv y))) = 0$.]

9. (a) False; $(x \oplus y) \lor z = (x \lor z) \oplus (y \lor z) \oplus z$. (b) True, because the identity obviously holds when $z = 0$ and when $z = 1$. (c) True; it's also $(x \oplus y) \lor (x \oplus z) = 1 - [x = y = z]$.

10. The first stage of decomposition (16) yields the functions with truth tables $g = 10100011$ and $h = 10100011 \oplus 10010011 = 00110000$; and the process continues in a similar way, yielding $1 + y + xz + w + wy + wx + wxz$ (modulo 2).

11. The stated term is present if and only if $f(x_1, \ldots, x_n)$ is true an odd number of times when $x_1 = x_4 = x_5 = x_7 = x_9 = x_{10} = \cdots = 0$. (There are 2^k such cases when we set all but k variables to zero.) In other words the multilinear representation can be expressed in a suggestive notation like

$$f(x, y, z) = (f_{000} + f_{00*}z + f_{0*0}y + f_{0**}yz + f_{*00}x + f_{*0*}xz + f_{**0}xy + f_{***}xyz) \bmod 2$$

illustrated here for $n = 3$, where $f_{**0} = f(1,1,0) \oplus f(1,0,0) \oplus f(0,1,0) \oplus f(0,0,0)$, etc.

12. (a) Substitute $1 - w$ for $\bar{w}$, etc., in (23), getting $1 - y - xz + 2xyz - w + wy + wx + wxz - 2wxyz$. [Some authors have called this the "Zhegalkin polynomial"; but I. I. Zhegalkin himself always worked modulo 2. Other names in the literature are "availability polynomial," "reliability polynomial," "characteristic polynomial."]

(b) The corresponding coefficients for an arbitrary n-ary function can be as large as 2^{n-1} in absolute value (and this, by induction, is the maximum). For example, the integer multilinear representation of $x_1 \oplus \cdots \oplus x_n$ over the integers turns out to be $e_1 - 2e_2 + 4e_3 - \cdots + (-2)^{n-1}e_n$, where e_k is the kth elementary symmetric function of $\{x_1, \ldots, x_n\}$. The formula in the previous answer becomes

$$f(x, y, z) \;=\; f_{000} + f_{00*}z + f_{0*0}y + f_{0**}yz + f_{*00}x + f_{*0*}xz + f_{**0}xy + f_{***}xyz$$

over the integers, where we now have $f_{**0} = f(1,1,0) - f(1,0,0) - f(0,1,0) + f(0,0,0)$, etc. This expansion is a disguised form of the Hadamard transform, Eq. 4.6.4–(38).

(c,d) The polynomial is the sum of its minterms like $x_1(1 - x_2)(1 - x_3)x_4$. Each minterm is nonnegative for $0 \le x_1, \ldots, x_n \le 1$, and the sum of all minterms is 1.

(e) $\partial f / \partial x_j = h(x) - g(x)$, where $h(x) \ge g(x)$ by (d). (See exercise 21.)

13. In fact, F is precisely the integer multilinear representation (see exercise 12).

14. Let $r_j = p_j/(1-p_j)$. We want $f(0,0,0) = 0$ and $f(1,1,1) = 1 \Leftrightarrow r_1r_2r_3 > 1$, $f(0,0,1) = 0$ and $f(1,1,0) = 1 \Leftrightarrow r_1r_2 > r_3$, $f(0,1,0) = 0$ and $f(1,0,1) = 1 \Leftrightarrow r_1r_3 > r_2$, $f(0,1,1) = 0$ and $f(1,0,0) = 1 \Leftrightarrow r_1 > r_2r_3$. So we get (a) $\langle x_1x_2x_3\rangle$; (b) x_1; (c) $\bar{x}_3$.

15. Exercise 1.2.6–10 tells us that $\binom{x}{k} \bmod 2 = [x \,\&\, k = k]$. Hence, for example, $\binom{x}{11} \equiv x_4 \wedge x_2 \wedge x_1$ (modulo 2) when $x = (x_n \ldots x_1)_2$; and we can obtain every term in a multilinear representation like (19) in this way. Moreover, we needn't work mod 2, because the interpolating polynomial $\binom{x}{11}\binom{15-x}{4}$ represents $x_4 \wedge \bar{x}_3 \wedge x_2 \wedge x_1$ exactly.

16. Yes, or even by +, because different minterms can't be simultaneously true. (But we can't do that in ordinary disjunctive normal forms like (25). See exercise 35.)

17. The binary operation $\overline{\wedge}$ is not associative, so an expression like $x \,\overline{\wedge}\, y \,\overline{\wedge}\, z$ must be interpreted as a *ternary* operation. Quick's notation is fine if one understands NAND to be an n-ary operation, being careful to note that the NAND of a *single* variable x is $\bar{x}$.

18. If not, we could set $u_1 \leftarrow \cdots \leftarrow u_s \leftarrow 1$ and $v_1 \leftarrow \cdots \leftarrow v_t \leftarrow 0$, making f both true and false. (And if we consider applying the distributive law (2) repeatedly to a DNF until it becomes a CNF, we find that the converse is also true: The disjunction $v_1 \vee \cdots \vee v_t$ is implied by f if and only if it has a literal in common with every implicant of f, if and only if it has a literal in common with every prime implicant of f, if and only if it has a literal in common with every implicant of some DNF for f.)

19. The maximal subcubes contained in 0010, 0011, 0101, 0110, 1000, 1001, 1010, and 1011 are 0*10, 0101, *01*, and 10**; so the answer is $(w\vee\bar{y}\vee z)\wedge(w\vee\bar{x}\vee y\vee\bar{z})\wedge(x\vee\bar{y})\wedge(\bar{w}\vee x)$. (This CNF is also shortest.)

20. True. The corresponding maximal subcube is contained in some maximal subcubes f' and g', and their intersection can't be larger. (This observation is due to Samson and Mills, whose paper is cited in answer 31 below.)

21. By Boole's law (20), we see that an n-ary function f is monotone if and only if its $(n-1)$-ary projections g and h are monotone and satisfy $g \le h$. Therefore

$$f = (g \wedge \bar{x}_n) \vee (h \wedge x_n) \;=\; (g \wedge \bar{x}_n) \vee (g \wedge x_n) \vee (h \wedge x_n) \;=\; g \vee (h \wedge x_n),$$

so we can do without complementation. The constants 0 and 1 disappear unless the function is identically constant. Conversely, any expression built up from $\wedge$ and $\vee$ is obviously monotone.

Note on terminology: Strictly speaking, we should say "monotone nondecreasing" instead of simply "monotone," if we want to preserve the language of classical mathematics, because a decreasing function of a real variable is also said to be monotonic. (See, for example, the "run test" in Section 3.3.2G.) But "nondecreasing" is quite a mouthful; so researchers who work extensively on Boolean functions have almost unanimously opted to assume that "monotone" automatically implies nondecreasing, in a Boolean context. Similarly, the mathematical term "positive function" normally refers to a function whose value exceeds zero; but authors who write about "positive Boolean functions" are referring to the functions that we are calling monotone. Since a monotone function is order-preserving, some authors have adopted the term *isotone*; but that word has already been coopted by physicists, chemists, and musicologists.

A Boolean function like $\bar{x} \vee y$, which becomes monotone if some subset of its variables is complemented, is called *unate*. Theorem Q obviously applies to unate functions.

22. Both g and $g \oplus h$ must be monotone, and $g(x) \wedge h(x) = 0$.

23. $x \wedge (v\vee y) \wedge (v\vee z) \wedge (w\vee z)$. (Corollary Q applies also to *conjunctive* prime forms of monotone functions. Therefore, to solve any problem of this kind, we need only

apply the distributive law (2) until no $\wedge$ occurs within a $\vee$, then remove any clause that contains all the variables of another.)

24. By induction on k, the similar tree with $\vee$ at the root gives a function with $2^{2^{\lceil k/2\rceil}-1}$ prime implicants of length $2^{\lfloor k/2\rfloor}$, while the tree with $\wedge$ gives $4^{2^{\lfloor k/2\rfloor}-1}$ of length $2^{\lceil k/2\rceil}$. When $k=6$, for example, the $4^7=2^{14}$ prime implicants in the $\wedge$ case have the form

$$x_{(0t_00t_{00}0t_{000})_2}\wedge x_{(0t_00t_{00}1t_{001})_2}\wedge x_{(0t_01t_{01}0t_{010})_2}\wedge x_{(0t_01t_{01}1t_{011})_2}$$
$$\wedge\, x_{(1t_10t_{10}0t_{100})_2}\wedge x_{(1t_10t_{10}1t_{101})_2}\wedge x_{(1t_11t_{11}0t_{110})_2}\wedge x_{(1t_11t_{11}1t_{111})_2},$$

with the t's either 0 or 1. [For further information about such Boolean functions, see D. E. Knuth and R. W. Moore, *Artificial Intelligence* **6** (1975), 293–326; V. Gurvich and L. Khachiyan, *Discrete Mathematics* **169** (1997), 245–248.]

25. Let a_n be the answer. Then $a_2=a_3=2$, $a_4=3$, and $a_n=a_{n-2}+a_{n-3}$ for $n>4$, because the prime implicants when $n>4$ are either $p_{n-2}\wedge x_{n-1}$ or $p_{n-3}\wedge x_{n-2}\wedge x_n$ for some prime implicant p_k in the k-variable case. (These prime implicants correspond to minimal vertex covers of the path graph P_n. They are *shellable*, in the sense of exercise 35, when listed in lexicographic order. We have $a_n=(7P_n+10P_{n+1}+P_{n+2})/23$ when P_n is the Perrin number of exercise 7.1.4–15.)

26. (a) Let $x_j=[j\in J]$. Then $f(x)=0$ and $g(x)=1$. (This fact was exercise 18.)

(b) Suppose, for example, that $k\in J\in\mathcal{G}$ and $k\notin\bigcup_{I\in\mathcal{F}}I$, and assume that test (a) has been passed. Let $x_j=[j\in J\text{ and }j\neq k]$. Then $f(x)=1$; and $g(x)=0$, because every $J'\in\mathcal{G}$ with $J'\neq J$ contains an element $\notin J$.

(c) Again assume that condition (a) has been ruled out. If, say, $|J|>|\mathcal{F}|$, let $x_j=[j$ is the smallest element of $I\cap J$, for some $I\in\mathcal{F}]$. Then $f(x)=1$, $g(x)=0$.

(d) Now we assume that $\bigcup_{I\in\mathcal{F}}I=\bigcup_{J\in\mathcal{G}}J$. Each $I\in\mathcal{F}$ stands for $2^{n-|I|}$ vectors where $f(x)=0$; similarly, each $J\in\mathcal{G}$ stands for $2^{n-|J|}$ vectors where $g(x)=1$. If the sum s is less than 2^n, we can compute $s=s_0+s_1$, where s_0 counts the contributions to s when $x_n=0$. If $s_0<2^{n-1}$, set $x_n\leftarrow 0$; otherwise $s_1<2^{n-1}$, so we set $x_n\leftarrow 1$. Then we set $n\leftarrow n-1$; eventually all x_j are known, and $f(x)=1$, $g(x)=0$.

27. Let $m=\min(\{|I|\mid I\in\mathcal{F}\}\cup\{|J|\mid J\in\mathcal{G}\})$ be the length of the shortest prime clause or implicant. Then $N\cdot 2^{n-m}\geq\sum_{I\in\mathcal{F}}2^{n-|I|}+\sum_{J\in\mathcal{G}}2^{n-|J|}\geq 2^n$; so we have $m\leq\lg N$. If, say, $|I|=m$, some index k must appear in at least $1/m$ of the members $J\in\mathcal{G}$, because each J intersects I. This observation proves the hint.

Now let $A(0)=A(1)=1$ and $A(v)=1+A(v-1)+A(\lfloor\rho v\rfloor)$ for $v>1$. Then $A(|\mathcal{F}||\mathcal{G}|)$ is an upper bound on the number of recursive calls (the number of times X1 is performed). Letting $B(v)=A(v)+1$, we have $B(v)=B(v-1)+B(\lfloor\rho v\rfloor)$ for $v>1$, hence $B(v)\leq B(v-k)+kB(\lfloor\rho v\rfloor)$ for $v>k$. Taking $k=v-\lfloor\rho v\rfloor$ shows that $B(v)\leq((1-\rho)v+2)B(\lfloor\rho v\rfloor)$; hence $B(v)=O(((1-\rho)v+2)^t)$ when $\rho^t v\leq 1$, namely when $t\geq\ln v/\ln(1/\rho)=\Theta((\log v)(\log N))$. Consequently $A(|\mathcal{F}||\mathcal{G}|)\leq A(N^2/4)=N^{O(\log N)^2}$.

In practice the algorithm will run much faster than the pessimistic bounds just derived. Since the prime clauses of a function are the prime implicants of its dual, this problem is essentially the same as verifying that one given DNF is the dual of another. Moreover, if we start with $f(x)=0$ and repeatedly find minimal x's where $f(x)=g(\bar{x})=0$, we can "grow" f until we've obtained the dual of g.

The ideas presented here are due to M. L. Fredman and L. Khachiyan, *J. Algorithms* **21** (1996), 618–628, who also presented refinements that reduce the running time to $N^{O(\log N/\log\log N)}$. No polynomial-time algorithm is known; yet the problem is unlikely to be NP-complete, because we can solve it in less-than-exponential time.

28. This result is obvious once understood, but the notations and terminology can make it confusing; so let's consider a concrete example: If, say, $y_1 = y_4 = y_6 = 1$ and the other y_k are zero, the function g is true if and only if the prime implicants p_1, p_4, and p_6 cover all the places where f is true. Thus we see that there is a one-to-one correspondence between every implicant of g and every DNF for f that contains only prime implicants p_j. In this correspondence, the prime implicants of g correspond to the "irredundant" DNFs in which no p_j can be left out.

Numerous refinements of this principle have been discussed by R. B. Cutler and S. Muroga, *IEEE Transactions* **C-36** (1987), 277–292.

29. B1. [Initialize.] Set $k \leftarrow k' \leftarrow 0$. (Similar methods are discussed in exercise 5–19.)

B2. [Find a zero.] Increase k zero or more times, until either $k = m$ (terminate) or $v_k \,\&\, 2^j = 0$.

B3. [Make $k' > k$.] If $k' \le k$, set $k' \leftarrow k + 1$.

B4. [Advance k'.] Increase k' zero or more times, until either $k' = m$ (terminate) or $v_{k'} \ge v_k + 2^j$.

B5. [Skip past a big mismatch.] If $v_k \oplus v_{k'} \ge 2^{j+1}$, set $k \leftarrow k'$ and return to B2.

B6. [Record a match.] If $v_{k'} = v_k + 2^j$, output (k, k').

B7. [Advance k.] Set $k \leftarrow k + 1$ and return to B2. ▌

(Steps B3 and B5 are optional, but recommended.)

30. The following algorithm keeps variable-length, sorted lists in a stack S whose size will never exceed $2m + n$. When the topmost entry of the stack is $S_t = s$, the topmost list is the ordered set $S_s < S_{s+1} < \cdots < S_{t-1}$. Tag bits are maintained in another stack T, having the same size as S (after the initialization step).

P1. [Initialize.] Set $T_k \leftarrow 0$ for $0 \le k < m$. Then for $0 \le j < n$, apply the j-buddy scan algorithm of exercise 29, and set $T_k \leftarrow T_k + 2^j$, $T_{k'} \leftarrow T_{k'} + 2^j$ for all pairs (k, k') found. Then set $s \leftarrow t \leftarrow 0$ and repeat the following operations until $s = m$: If $T_s = 0$, output the subcube $(0, v_s)$ and set $s \leftarrow s+1$; otherwise set $S_t \leftarrow v_s$, $T_t \leftarrow T_s$, $t \leftarrow t + 1$, $s \leftarrow s + 1$. Finally set $A \leftarrow 0$ and $S_t \leftarrow 0$.

P2. [Advance A.] (At this point stack S contains $\nu(A) + 1$ lists of subcubes. Namely, if $A = 2^{e_1} + \cdots + 2^{e_r}$ with $e_1 > \cdots > e_r \ge 0$, the stack contains the b-values of all subcubes $(a, b) \subseteq V$ whose a-values are respectively 0, 2^{e_1}, $2^{e_1} + 2^{e_2}, \ldots, A$, except that subcubes whose tags are zero do not appear. All of these lists are nonempty, except possibly the last. We will now increase A to the next relevant value.) Set $j \leftarrow 0$. If $S_t = t$ (that is, if the topmost list is empty), increase j zero or more times until $j \ge n$ or $A \,\&\, 2^j \ne 0$. Then while $j < n$ and $A \,\&\, 2^j \ne 0$, set $t \leftarrow S_t - 1$, $A \leftarrow A - 2^j$, and $j \leftarrow j + 1$. Terminate the algorithm if $j \ge n$; otherwise set $A \leftarrow A + 2^j$.

P3. [Generate list A.] Set $r \leftarrow t$, $s \leftarrow S_t$, and apply the j-buddy scan algorithm of exercise 29 to the $r - s$ numbers $S_s < \cdots < S_{r-1}$. For all pairs (k, k') found, set $x \leftarrow (T_k \,\&\, T_{k'}) - 2^j$; and if $x = 0$, output the subcube (A, S_k), otherwise set $t \leftarrow t + 1$, $S_t \leftarrow S_k$, $T_t \leftarrow x$. Finally set $t \leftarrow t + 1$, $S_t \leftarrow r + 1$, and go back to step P2. ▌

This algorithm is based in part on ideas of Eugenio Morreale [*IEEE Trans.* **EC-16** (1967), 611–620; *Proc. ACM Nat. Conf.* **23** (1968), 355–365]. The running time is essentially proportional to mn (for step P1) plus the total number of subcubes contained in V. If $m \le 2^n(1 - \epsilon)$, and if V is chosen at random with size m, exercise 34 shows that the average total number of subcubes is at most $O(\log\log n / \log\log\log n)$ times

the average number of maximal subcubes; hence the average running time in most cases will be nearly proportional to the average amount of output produced. On the other hand, exercises 32 and 116 show that the amount of output might be huge.

31. (a) Let $c = c_{n-1}\ldots c_0$, $c' = c'_{n-1}\ldots c'_0$, $c'' = c''_{n-1}\ldots c''_0$. There must be some j with $c_j \neq *$ and $c_j \neq c''_j$; otherwise $c'' \subseteq c$. Similarly there must be some k with $c'_k \neq *$ and $c'_k \neq c''_k$. If $j \neq k$, there would be a point $x_{n-1}\ldots x_0 \in c''$ that is in neither c nor c', because we could let $x_j = \bar{c}_j$ and $x_k = \bar{c}'_k$. Hence $j = k$, and the value of j is uniquely determined. Furthermore it's easy to see that $c'_j = \bar{c}_j$. And if $i \neq j$, we have either $c_i = *$ or $c_i = c''_i$, and either $c'_i = *$ or $c'_i = c''_i$.

(b) This statement is an obvious consequence of (a).

(c) First we prove that the parenthesized remark in step E2 is true whenever that step is encountered. It's clearly true when $j = 0$. Otherwise, let $c \subseteq V$ be a j-cube, and suppose $c = c_0 \cup c_1$ where c_0 and c_1 are $(j-1)$-cubes. On the preceding execution of step E2 we had $c_0 \subseteq c'_0 \in C$ and $c_1 \subseteq c'_1 \in C$ for some c'_0 and c'_1; hence either $c \subseteq c'_0 \sqcup c'_1$ or $c \subseteq c'_0$ or $c \subseteq c'_1$. In each case, c is now contained in some element of C.

Secondly, we prove that the outputs in step E3 are precisely the maximal j-cubes contained in V: Let $c \subseteq V$ be any k-cube. If c is maximal, then c will be in C when we reach step E3 with $j = k$, and it will be output. If c isn't maximal, it has a buddy $c' \subseteq V$, which is a k-cube contained in some subcube $c'' \in C$ when we reach E3. Since $c \not\subseteq c''$, the consensus $c \sqcup c''$ will be a $(j+1)$-cube of C', and c will not be output.

References: The notion of consensus was first defined by Archie Blake in his Ph.D. dissertation at the University of Chicago (1937); see *J. Symbolic Logic* **3** (1938), 93, 112–113. It was independently rediscovered by Edward W. Samson and Burton E. Mills [Air Force Cambridge Research Center Tech. Report 54-21 (Cambridge, Mass.: April 1954), 54 pp.] and by W. V. Quine [*AMM* **62** (1955), 627–631]. The operation is also sometimes called the *resolvent*, since J. A. Robinson used it in a more general form (but with respect to clauses rather than implicants) as the basis of his "resolution principle" for theorem proving [*JACM* **12** (1965), 23–41]. Algorithm E is due to Ann C. Ewing, J. Paul Roth, and Eric G. Wagner, *AIEE Transactions*, Part 1, **80** (1961), 450–458.

32. (a) Change the definition of $\sqcup$ in exercise 31 to the following associative and commutative operation on the four symbols $A = \{0, 1, *, \bullet\}$, for all $a \in A$ and $x \in \{0, 1\}$:

$$* \sqcup a = a \sqcup * = a, \qquad \bullet \sqcup a = a \sqcup \bullet = x \sqcup \bar{x} = \bullet, \qquad \text{and} \qquad x \sqcup x = x.$$

Also let $h(0) = 0$, $h(1) = 1$, $h(*) = *$, and $h(\bullet) = *$. Then $c = h(c_1 \sqcup \cdots \sqcup c_m)$, computed componentwise, is the generalized consensus if and only if this subcube is contained in $c_1 \cup \cdots \cup c_m$. [See P. Tison, *IEEE Transactions* **EC-16** (1967), 446–456.]

(b) For example, let $c_j = *^{j-1}1*^{m-j}1^{j-1}0*^{m-j}$. [The final component is superfluous. All solutions have been characterized by R. H. Sloan, B. Szörényi, and G. Turán, in *Electronic Colloquium on Computational Complexity* (2005), Report 23.]

(c) By (a), every prime implicant corresponds uniquely to the subset of implicants that it "meets." [A. K. Chandra and G. Markowsky, *Discrete Math.* **24** (1978), 7–11.]

(d) For example, $(y_1\wedge\bar{x}_1) \vee (y_2\wedge x_1\wedge\bar{x}_2) \vee \cdots \vee (y_m\wedge x_1\wedge\cdots\wedge x_{m-1}\wedge\bar{x}_m)$ as in (b). [J.-M. Laborde, *Discrete Math.* **32** (1980), 209–212.]

33. (a) $\binom{2^n-2^{n-k}}{m-2^{n-k}}/\binom{2^n}{m}$. (b) We must exclude the cases when $x_1 \wedge \cdots \wedge x_{j-1} \wedge \bar{x}_j \wedge x_{j+1} \wedge \cdots \wedge x_k$ is also an implicant. By the inclusion-exclusion principle, the answer is

$$\sum_l \binom{k}{l}(-1)^l \binom{2^n-(l+1)2^{n-k}}{m-(l+1)2^{n-k}} \Big/ \binom{2^n}{m};$$

it simplifies to $\binom{2^n-n-1}{m-1}/\binom{2^n}{m}$ when $k = n$.

34. (a) We have $c(m,n)=\sum c_j(m,n)$, where $c_j(m,n)=2^{n-j}\binom{n}{j}\binom{2^n-2^j}{m-2^j}/\binom{2^n}{m}$ is the average number of implicants with $n-j$ literals (the average number of subcubes of dimension j in the terminology of exercise 30). Clearly $c_0(m,n)=m$, and

$$c_1(m,n)=\frac{nm(m-1)}{2(2^n-1)}\le\frac{mn}{2}\left(\frac{m}{2^n}\right)\le\frac{1}{2}m;$$

similarly $c_j(m,n)\le m/(2^j j!\,n^{2^j-1-j})$. Also $p(m,n)=\sum_j p_j(m,n)$, where we have

$$p_0(m,n)=2^n\binom{2^n-n-1}{m-1}\Big/\binom{2^n}{m}=m\frac{(2^n-n-1)^{\underline{m-1}}}{(2^n-1)^{\underline{m-1}}}\ge m\frac{(2^n-n-m)^{m-1}}{(2^n-m)^{m-1}}$$
$$\ge m\left(1-\frac{n}{2^n-m}\right)^m\ge m\left(1-\frac{n}{2^n-2^n/n}\right)^{2^n/n}=m\exp\left(\frac{2^n}{n}\ln\left(1-\frac{n^2}{2^n(n-1)}\right)\right).$$

(b) Notice that $t=\lfloor\lg\lg n-\lg\lg(2^n/m)+\lg(4/3)\rfloor\le\lg\lg n+O(1)$ is quite small. We will repeatedly use the fact that $\binom{2^n-j\cdot2^t}{m-j\cdot2^t}/\binom{2^n}{m}<\alpha_{mn}^j$, and indeed that

$$\binom{2^n-j\cdot2^t}{m-j\cdot2^t}\Big/\binom{2^n}{m}\;=\;\alpha_{mn}^j(1+O(j^2 2^{2t}/m))$$

is an extremely good approximation when j isn't too large. To establish the hint, note that $\sum_{j<t}c_j(m,n)/c_t(m,n)=O(tc_{t-1}(m,n)/c_t(m,n))=O(t^2/(n\sqrt{\alpha_{mn}}))=O((\log\log n)^2/n^{1/3})$; and $c_{t+j}(m,n)/c_t(m,n)=O((n/(2t))^j\alpha_{mn}^{2^j-1})$. Consequently we have $c(m,n)/c_t(m,n)\approx1+\frac12\binom{n-t}{t+1}\alpha_{mn}$, where the second term dominates when α_{mn} is in the upper part of its range. Furthermore

$$\sum_l\binom{n-t}{l}(-1)^l\alpha_{mn}^l\left(1+O\left(\frac{l^2 2^{2t}}{m}\right)\right)\;=\;(1-\alpha_{mn})^{n-t}+O(n^2\alpha_{mn}(1+\alpha_{mn})^n 2^{2t}/m)$$

has an exponentially small error term, because $(1+\alpha_{mn})^n=O(e^{n^{1/3}})\ll m$. Therefore $p(m,n)/c_t(m,n)$ is asymptotically $e^{-n\alpha_{mn}}+\frac12\binom{n-t}{t+1}\alpha_{mn}e^{-n\alpha_{mn}^2}$.

(c) Here $\alpha_{mn}=2^{-2^t}\approx n^{-1}\ln(t/\ln t)$; so $c(m,n)/c_t(m,n)=1+O(t^{-1}\log t)$, $p(m,n)/c_t(m,n)=t^{-1}\ln t+\frac12t^{-1}\ln t+O(t^{-1}\log\log t)$. We conclude that, in this case,

$$\frac{c(m,n)}{p(m,n)}\;=\;\frac{2}{3}\frac{\lg\lg n}{\lg\lg\lg n}\left(1+O\left(\frac{\log\log\log\log n}{\log\log\log n}\right)\right).$$

(d) If $n\alpha_{mn}\le\ln t-\ln\ln t$, we have $p(m,n)/c(m,n)\ge p_t(m,n)/c(m,n)\ge t^{-1}\ln t+O(t^{-1}\log t)^2$. On the other hand if $n\alpha_{mn}\ge\ln t-\ln\ln t$, we have $p(m,n)/c(m,n)\ge p_{t+1}(m,n)/c(m,n)\ge\frac12t^{-1}\ln t+O(t^{-1}\log\log t)$.

[The means $c(m,n)$ and $p(m,n)$, and the variance of $c(m,n)$, were first studied by F. Mileto and G. Putzolu, *IEEE Trans.* **EC-13** (1964), 87–92; *JACM* **12** (1965), 364–375. Detailed asymptotic information about implicants, prime implicants, and irredundant DNFs of random Boolean functions, when each value $f(x_1,\ldots,x_n)$ is independently equal to 1 with probability $p(n)$, has been derived by Karl Weber, *Elektronische Informationsverarbeitung und Kybernetik* **19** (1983), 365–374, 449–458, 529–534.]

35. (a) By rearranging coordinates we can assume that the pth subcube is $0^k1^u{*}^v$, so that $B_p=0^k1^u0^v$ and $S_p=1^k0^{u+v}$. Then all points of ${*}^k1^u{*}^v$ are still covered, by induction on p, because all points of ${*}^{j-1}1{*}^{k-j}1^u{*}^v$ have been covered for $1\le j\le k$.

(b) The jth and kth subcubes differ in every coordinate position where $B_j\,\&\,S_k$ is nonzero. On the other hand if $B_j\,\&\,S_k$ is zero, the point $\bar S_k$ of subcube k lies in a previous subcube, by (a), because we have $\bar S_k\supseteq B_j$.

(c) From the list 1100, $1\underline{0}11$, $\underline{0}011$ (with the bits of each S_k underlined) we obtain the orthogonal DNF $(x_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4) \vee (\bar{x}_1 \wedge x_3 \wedge x_4)$.

(d) There are eight solutions; for example, $(01100, 0\underline{0}110, 00\underline{0}11, 11\underline{0}1\underline{0}, 11\underline{00}0)$.

(e) $(001100, 011\underline{0}00, 00\underline{0}110, 11\underline{00}10, 11\underline{0}0\underline{0}0, \underline{0}1\underline{00}11, 0\underline{0}0\underline{0}11)$ is a symmetrical solution. And there are many more possibilities; for example, 42 permutations of the bit codes $\{110000, 011000, 001100, 000110, 000011, 110010, 011010\}$ are shellings.

[The concept of a shelling for monotone Boolean functions was introduced by Michael O. Ball and J. Scott Provan, *Operations Research* **36** (1988), 703–715, who discussed many significant applications.]

36. If $j < k$ we have $B_j = \alpha 1 \beta$ and $B_k = \alpha 0 \gamma$ for some strings α, β, γ. Form the sequence $x_0 = \alpha 1 \gamma$, $x_1 = x_0'$, …, $x_l = x_{l-1}'$, where $x_l = \alpha 0 0^{|\gamma|}$. We have $f(x_0) = 1$ since $x_0 \supseteq B_k$, but $f(x_l) = 0$ since $x_l \subseteq B_j'$. So the string x_i, where $f(x_i) = 1$ and $f(x_{i+1}) = \cdots = f(x_l) = 0$, is in B. It precedes B_k and proves that $B_j \,\&\, S_k \supseteq 0^{|\alpha|} 1 0^{|\beta|}$.

[This construction and parts of exercise 35 are due to E. Boros, Y. Crama, O. Ekin, P. L. Hammer, T. Ibaraki, and A. Kogan, *SIAM J. Discrete Math.* **13** (2000), 212–226.]

37. The shelling order $(000011, 0011\underline{0}1, 00110\underline{0}, 11\underline{0}1\underline{0}1, 11\underline{0}10\underline{0}, 110\underline{00}1, 110\underline{0}0\underline{0})$ generalizes to all n. There also are interesting solutions not based on shelling, like the cyclically symmetrical (110***, 1110**, **110*, **1110, 0***11, 10**11, 111111).

For the lower bound, assign the weight $w_x = -\prod_{j=1}^{n}(x_{2j-1} + x_{2j} - 3x_{2j-1}x_{2j})$ to each point x, and notice that the sum of w_x over all x in any subcube is 0 or ± 1. (It suffices to verify this curious fact for each of the nine possible subcubes when $n = 1$.) Now choose a set of disjoint subcubes that partition the set $F = \{x \mid f(x) = 1\}$; we have

$$\sum_{C \text{ chosen}} 1 \;\geq\; \sum_{C \text{ chosen}} \sum_{x \in C} w_x \;=\; \sum_{x \in F} w_x \sum_{C \text{ chosen}} [x \in C] \;=\; \sum_{x \in F} w_x.$$

There are $\binom{n}{k} 2^{n-k}$ vectors x with exactly k pairs $x_{2j-1}x_{2j} = 1$ and nonzero weight. Their weight is $(-1)^{k-1}$, and they lie in F except when $k = 0$. Hence $\sum_{x \in F} w_x = \sum_{k > 0} \binom{n}{k} 2^{n-k} (-1)^{k-1} = 2^n - (2-1)^n$. [See M. O. Ball and G. L. Nemhauser, *Mathematics of Operations Research* **4** (1979), 132–143.]

38. Certainly not; a DNF is satisfiable if and only if it has at least one implicant. The hard problem for a DNF is to decide whether or not it is a *tautology* (always true).

39. Associate variables y_1, …, y_N with each internal node in preorder, so that every tree node corresponds to exactly one variable of F. For each internal node y, with children (l, r) and labeled with the binary operator $\circ$, construct four 3CNF clauses $c_{00} \wedge c_{01} \wedge c_{10} \wedge c_{11}$, where

$$c_{pq} = (y^{\overline{p \circ q} N} \vee l^{pN} \vee r^{qN})$$

and N denotes complementation (so that $x^{0N} = x$ and $x^{1N} = \bar{x}$). These clauses state in effect that $y = l \circ r$; for example, if $\circ$ is $\wedge$, the four clauses are $(\bar{y} \vee l \vee r) \wedge (\bar{y} \vee l \vee \bar{r}) \wedge (\bar{y} \vee \bar{l} \vee r) \wedge (y \vee \bar{l} \vee \bar{r})$. Finally, add one more clause, $(y_1 \vee y_1 \vee y_1)$, to force $F = 1$.

Every higher number can be formed by mere complications of threes.
... Take the quadruple fact that A sells B to C for the price D.
This is a compound of two facts:
first, that A makes with C a certain transaction, which we may name E;
and second, that this transaction E is a sale of B for the price D.

— CHARLES S. PEIRCE, *A Guess at the Riddle* (1887)

40. Following the hint, A says '$u < v \oplus v < u$' and B says '$u < v \wedge v < w \Rightarrow u < w$'. So $A \wedge B$ says that there's a linear ordering of the vertices, $u_1 < u_2 < \cdots < u_n$. (There are $n!$ ways to satisfy $A \wedge B$.) Now C says that q_{uvw} is equivalent to $u < v < w$; so D says that u and w are not consecutive in the ordering, when $u \not\!\!- w$. Thus $A \wedge B \wedge C \wedge D$ is satisfiable if and only if there is a linear ordering in which all nonadjacent vertices are nonconsecutive (that is, in which all consecutive vertices are adjacent).

41. Solution 0: '$[m \le n]$' is such a formula, but it is not in the spirit of this exercise.

Solution 1: Let x_{jk} mean that pigeon j occupies hole k. Then the clauses are $(x_{j1} \vee \cdots \vee x_{jn})$ for $1 \le j \le m$ and $(\bar{x}_{ik} \vee \bar{x}_{jk})$ for $1 \le i < j \le m$ and $1 \le k \le n$. [See S. A. Cook and R. A. Reckhow, *J. Symbolic Logic* **44** (1979), 36–50; A. Haken, *Theoretical Comp. Sci.* **39** (1985), 297–308.]

Solution 2: Assume that $n = 2^t$ and let pigeon j occupy hole $(x_{j1} \ldots x_{jt})_2$. The clauses $((x_{i1} \oplus x_{j1}) \vee \cdots \vee (x_{it} \oplus x_{jt}))$ for $1 \le i < j \le m$ can be put into the CNF form $(y_{ij1} \vee \cdots \vee y_{ijt})$ as in exercise 39, by introducing auxiliary clauses $(\bar{y}_{ijk} \vee x_{ik} \vee x_{jk}) \wedge (y_{ijk} \vee x_{ik} \vee \bar{x}_{jk}) \wedge (y_{ijk} \vee \bar{x}_{ik} \vee x_{jk}) \wedge (\bar{y}_{ijk} \vee \bar{x}_{ik} \vee \bar{x}_{jk})$. The total size of this CNF is $\Theta(m^2 \log n)$, compared to $\Theta(m^2 n)$ in Solution 1. If n is not a power of 2, $O(m \log n)$ additional clauses of size $O(\log n)$ will rule out inappropriate values.

42. $(\bar{x} \vee y) \wedge (\bar{z} \vee x) \wedge (\bar{y} \vee \bar{z}) \wedge (z \vee z)$.

43. Probably not, because every 3SAT problem can be converted to this form. For example, the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ can be replaced by $(x_1 \vee \bar{y} \vee \bar{x}_3) \wedge (\bar{y} \vee \bar{x}_2) \wedge (y \vee x_2)$, where y is a new variable (essentially equivalent to $\bar{x}_2$).

44. Suppose $f(x) = f(y) = 1$ implies $f(x\&y) = 1$ and also that, say, $c = x_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$ is a prime clause of f. Then $c' = \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$ is *not* a clause; otherwise $c \wedge c' = x_2 \vee \bar{x}_3 \vee \bar{x}_4$ would also be a clause, contradicting primality. So there's a vector y with $f(y) = 1$ and $y_1 = 1$, $y_2 = 0$, $y_3 = y_4 = 1$. Similarly, there's a z with $f(z) = 1$ and $z_1 = 0$, $z_2 = 1$, $z_3 = z_4 = 1$. But then $f(y \,\&\, z) = 1$, and c isn't a clause. The same argument works for a clause c that has a different number of literals, as long as at least two of the literals aren't complemented.

45. (a) A Horn function $f(x_1, \ldots, x_n)$ is indefinite if and only if it is unequal to the definite Horn function $g(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$. So $f \leftrightarrow g$ is a one-to-one correspondence between indefinite and definite Horn functions. (b) If f is monotone, its complement $\bar{f}$ is either identically 1 or an indefinite Horn function.

46. Algorithm C puts 88 pairs `xy` in the core: When `x` = `a`, `b`, `c`, `0`, or `1`, the following character `y` can be anything but `(`. When `x` = `(`, `*`, `/`, `+`, `-`, we can have `y` = `(`, `a`, `b`, `c`, `0`, `1`; also `y` = `-` when `x` = `(`, `+`, or `-`. Finally, the legitimate pairs beginning with `x` = `)` are `)+`, `)-`, `)*`, `)/`, `))`.

47. The order in which Algorithm C brings vertices into the core is a topological sort, since all predecessors of k are asserted before the algorithm sets `TRUTH`$(x_k) \leftarrow 1$. But Algorithm 2.2.3T uses a queue instead of a stack, so the ordering it actually produces is usually different from that of Algorithm C.

48. Let $\bot$ be a new variable, and change every indefinite Horn clause to a definite one by ORing in this new variable. (For example, '$\bar{w} \vee \bar{y}$' becomes '$\bar{w} \vee \bar{y} \vee \bot$', namely '$w \wedge y \Rightarrow \bot$'; definite Horn clauses stay unchanged.) Then apply Algorithm C. The original clauses are unsatisfiable if and only if $\bot$ is in the core of the new clauses. The algorithm can therefore be terminated as soon as it is about to set `TRUTH`$(\bot) \leftarrow 1$.

(J. H. Quick thought of another solution: We could apply Algorithm C to the function g constructed in the answer to exercise 45(a), because f is unsatisfiable if and

only if *every* variable x_j is in the core of g. However, indefinite clauses of f such as $\bar{w} \vee \bar{y}$ become many different clauses $(\bar{w} \vee \bar{y} \vee z) \wedge (\bar{w} \vee \bar{y} \vee x) \wedge (\bar{w} \vee \bar{y} \vee v) \wedge (\bar{w} \vee \bar{y} \vee u) \wedge \cdots$ of g, one for each variable not in the original clause. So Quick's suggestion, which might sound elegant at first blush, could increase the number of clauses by a factor of $\Omega(n)$.)

49. We have $f \le g$ if and only if $f \wedge \bar{g}$ is unsatisfiable, if and only if $f \wedge \bar{c}$ is unsatisfiable for every clause c of g. But $\bar{c}$ is an AND of literals, so we can apply exercise 48. [See H. Kleine Büning and T. Lettmann, *Aussagenlogik: Deduktion und Algorithmen* (1994), §5.6, for further results including an efficient way to test if g is a "renaming" of f, namely to determine whether or not there exist constants $(y_1, \ldots, y_n)$ such that $f(x_1, \ldots, x_n) = g(x_1 \oplus y_1, \ldots, x_n \oplus y_n)$.]

50. See Gabriel Istrate, *Random Structures & Algorithms* **20** (2002), 483–506.

51. If vertex v is marked A, introduce the clauses $\Rightarrow A^+(v)$ and $\Rightarrow B^-(v)$; if it is marked B, introduce $\Rightarrow A^-(v)$ and $\Rightarrow B^+(v)$. Otherwise let v have k outgoing arcs $v \to u_1, \ldots, v \to u_k$. Introduce the clauses $A^-(u_j) \Rightarrow B^+(v)$ and $B^-(u_j) \Rightarrow A^+(v)$ for $1 \le j \le k$. Also, if v is not marked C, introduce the clauses $A^+(u_1) \wedge \cdots \wedge A^+(u_k) \Rightarrow B^-(v)$ and $B^+(u_1) \wedge \cdots \wedge B^+(u_k) \Rightarrow A^-(v)$. All forcing strategies are consequences of these clauses. Exercise 2.2.3–28 and its answer provide further information.

Notice that, in principle, Algorithm C can therefore be used to decide whether or not the game of chess is a forced victory for the white pieces — except for the annoying detail that the corresponding digraph is larger than the physical universe.

52. With best play, the results (see exercise 51) are:

n	(a)	(b)	(c)	(d)
2	0 wins	second player wins	1 wins	second player wins
3	0 wins	first player wins	first player wins	first player wins
4	first player wins	first player wins	first player wins	first player wins
5	second player wins	draw	draw	1 loses if first
6	second player wins	second player wins	1 loses if first	1 loses if first
7	1 loses if first	second player wins	1 loses if first	1 loses if first
8	draw	draw	draw	1 loses if first
9	draw	draw	draw	1 loses if first

(Here "1 loses if first" means that the game is a draw if player 0 plays first, otherwise 0 can win.) *Comments:* In (a), player 1 has a slight disadvantage, because $f(x) = 0$ when $x_1 \ldots x_n$ is a palindrome. This small difference affects the result even when $n = 7$. Although player 1 would seem to be better off playing 0s in the left half of the board, it turns out that his/her first move when $n = 4$ must be to *1**; the alternative, *0**, draws. Game (b) is essentially a race to see who can eliminate the last *. In game (c), a random choice of $x_1 \ldots x_n$ makes $f(x) = 1$ with probability $F_{n+2}/2^n = \Theta((\phi/2)^n)$; in game (d), this probability approaches zero more slowly, as $\Theta(1/\log n)$. Still, player 1 does better in (c) than in (d) when $n = 2$, 5, 8, and 9; no worse in the other cases.

53. (a) She should switch either day 1 or day 2 to day 3. (b, f) Several possibilities; for example, change day 2 to day 3. (c) This case is illustrated in Fig. 6; change either Desert or Excalibur to Aladdin. (d) Change either Caesars or Excalibur to Aladdin. (e) Change either Bellagio or Desert to Aladdin. Of course Williams, who doesn't appear in the cycle (42), bears no responsibility whatever for the conflicts.

54. If x and $\bar{x}$ are both in S, then $u \in S \iff \bar{u} \in S$, because the existence of paths from x to $\bar{x}$ and $\bar{x}$ to x and x to u and u to x implies the existence of paths from $\bar{u}$ to $\bar{x}$ and $\bar{x}$ to $\bar{u}$, hence from u to $\bar{u}$ and $\bar{u}$ to u.

55. (a) Necessary and sufficient conditions for successfully renaming a clause such as $x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4$ are $(y_1 \vee \bar{y}_2) \wedge (y_1 \vee y_3) \wedge (y_1 \vee \bar{y}_4) \wedge (\bar{y}_2 \vee y_3) \wedge (\bar{y}_2 \vee \bar{y}_4) \wedge (y_3 \vee \bar{y}_4)$. A similar set of $\binom{k}{2}$ clauses of length 2 in the variables $\{y_1, \ldots, y_n\}$ corresponds to any clause of length k in $\{x_1, \ldots, x_n\}$. [H. R. Lewis, *JACM* **25** (1978), 134–135.]

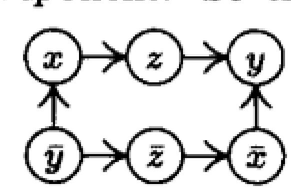
(b) A given clause of length $k > 3$ in $\{x_1, \ldots, x_n\}$ can be converted into $3(k-2)$ clauses of length 2, instead of the $\binom{k}{2}$ clauses above, by introducing $k-3$ new variables $\{t_2, \ldots, t_{k-2}\}$, illustrated here for the clause $x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$:

$$(y_1 \vee y_2) \wedge (y_1 \vee t_2) \wedge (y_2 \vee t_2) \wedge (\bar{t}_2 \vee y_3) \wedge (\bar{t}_2 \vee t_3) \wedge (y_3 \vee t_3) \wedge (\bar{t}_3 \vee y_4) \wedge (\bar{t}_3 \vee y_5) \wedge (y_4 \vee y_5).$$

In general, the clauses from $x_1 \vee \cdots \vee x_k$ are $(\bar{t}_{j-1} \vee y_j) \wedge (\bar{t}_{j-1} \vee t_j) \wedge (y_j \vee t_j)$ for $1 < j < k$, but with t_1 replaced by $\bar{y}_1$ and t_{k-1} replaced by y_k; change y_j to $\bar{y}_j$ if $\bar{x}_j$ appears instead of x_j. Do this for each given clause, using different auxiliary variables t_j for different clauses; the result is a formula in 2CNF that has length $< 3m$ and is satisfiable if and only if Horn renaming is possible. Now apply Theorem K.

[See B. Aspvall, *J. Algorithms* **1** (1980), 97–103. One consequence, noted by H. Kleine Büning and T. Lettmann in *Aussagenlogik: Deduktion und Algorithmen* (1994), Theorem 5.24, is that any satisfiable formula in 2CNF can be renamed to Horn clauses. Notice that two CNFs for the same function may give different outcomes; for example, $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee z)$ is actually a Horn function, but the clauses in this representation cannot be converted to Horn form by complementation.]

56. Here $f(x, y, z)$ corresponds to the digraph shown below (analogous to Fig. 6), and it can also be simplified to $y \wedge (\bar{x} \vee z)$. Each vertex is a strong component. So the formula is true with respect to the quantifiers $\exists\exists\exists$, $\exists\exists\forall$, $\forall\exists\exists$; false in the other cases $\forall\exists\forall$, (any)$\forall$(any). In general the eight possibilities can be arranged at the corners of a cube, with each change from $\exists$ to $\forall$ making the formula more likely to be false.



57. Forming the digraph as in Theorem K, we can prove that the quantified formula holds if and only if (i) no strong component contains both x and $\bar{x}$; (ii) there is no path from one universal variable x to another universal variable y or to its complement $\bar{y}$; (iii) no strong component containing a universal variable x also contains an existential variable v or its complement $\bar{v}$, when '$\exists v$' appears to the left of '$\forall x$'. These three conditions are clearly necessary, and they are readily tested as the strong components are being found.

To show that they are sufficient, notice first that if S is a strong component with only existential literals, condition (i) allows us to set them all equal as in Theorem K. Otherwise S has exactly one universal literal, $u_j = x_j$ or $u_j = \bar{x}_j$; all other literals in S are existential and declared to the right of x_j, so we can equate them to u_j. And all paths into S in such a case come from purely existential strong components, whose value can be set to 0 because the complements of such strong components cannot also lead into S; for if v and $\bar{v}$ imply u_j, then $\bar{u}_j$ implies $\bar{v}$ and v.

[*Information Proc. Letters* **8** (1979), 121–123. By contrast, M. Krom had proved in *J. Symbolic Logic* **35** (1970), 210–216, that an analogous problem in first-order predicate calculus (where parameterized predicates take the place of simple Boolean variables, and quantification is over the parameters) is actually unsolvable in general.]

58. We can assume that each clause is definite, by introducing '$\bot$' as in exercise 48 and placing '$\forall\bot$' at the left. Call the universal variables $x_0, x_1, \ldots, x_m$ (where x_0 is $\bot$) and call the existential variables $y_1, \ldots, y_n$. Let '$u \prec v$' mean that variable u appears to the left of variable v in the list of quantifiers. Remove $\bar{x}_j$ from any clause whose unbarred

literal is y_k when $y_k \prec x_j$. Then, for $0 \le j \le m$, let C_j be the core of the Horn clauses when the additional clauses $(x_0) \wedge \cdots \wedge (x_{j-1}) \wedge (x_{j+1}) \wedge \cdots \wedge (x_m) \wedge \bigwedge\{(y_k) \mid y_k \prec x_j$ and $y_k \in C_0\}$ are appended. (In other words, C_j tells us what can be deduced when all the x's except x_j are assumed to be true.) We claim that the given formula is true if and only if $x_j \notin C_j$, for $0 \le j \le m$.

To prove this claim, note first that the formula is certainly false if $x_j \in C_j$ for some j. (When $y_k \in C_0$ and $y_k \prec x_j$ and $x_i = 1$ for $i \ne j$ we must set $y_k \leftarrow 1$.) Otherwise we can choose each y_k to make the formula true, as follows: If $y_k \notin C_0$, set $y_k \leftarrow 0$; otherwise set $y_k \leftarrow \bigwedge\{x_j \mid y_k \notin C_j\}$. Notice that y_k depends on x_j only when $x_j \prec y_k$. Each clause c with unbarred literal x_j is now true: For if $x_j = 0$, some $\bar{y}_k$ appears in c for which $y_k \notin C_j$, because $x_j \notin C_j$; hence $y_k = 0$. And each clause c with unbarred literal y_k is also true: If $y_k = 0$, we either have $y_k \notin C_0$, in which case some $\bar{y}_l$ in c is $\notin C_0$, hence $y_l = 0$; or $y_k \in C_0 \setminus C_j$ for some j, in which case some $x_j = 0$ and either $\bar{x}_j$ appears in c or some $\bar{y}_l$ appears in c where $y_l \notin C_j$, making $y_l = 0$.

[This solution is due to T. Dahlheimer. See M. Karpinski, H. Kleine Büning, and P. H. Schmitt, *Lecture Notes in Comp. Sci.* **329** (1988), 129–137; H. Kleine Büning, K. Subramani, and X. Zhao, *Lecture Notes in Comp. Sci.* **2919** (2004), 93–104.]

59. By induction on n: Suppose $f(0, x_2, \ldots, x_n)$ leads to the quantified results $y_1, \ldots, y_{2^n-1}$, while $f(1, x_2, \ldots, x_n)$ leads similarly to $z_1, \ldots, z_{2^n-1}$. Then $\exists x_1 f(x_1, x_2, \ldots, x_n)$ leads to $y_1 \vee z_1, \ldots, y_{2^n-1} \vee z_{2^n-1}$, and $\forall x_1 f(x_1, x_2, \ldots, x_n)$ leads to $y_1 \wedge z_1, \ldots, y_{2^n-1} \wedge z_{2^n-1}$. Now use the fact that $(y \vee z) + (y \wedge z) = y + z$. [See *Proc. Mini-Workshop on Quantified Boolean Formulas* **2** (QBF-02) (Cincinnati: May 2002), 1–16.]

60. Both (a) and (b). But (c) is always 0; (d) is always 1; (e) is $\overline{\langle xyz \rangle}$; (f) is $\bar{x} \vee \bar{y} \vee \bar{z}$.

61. True—indeed obviously so, when $w = 0$, and when $w = 1$.

62. Since $\{x_1, x_2, x_3\} \subseteq \{0, 1\}$, we can assume by symmetry that x_1 equals x_2. Then either $f(x_1, x_1, x_3, x_4, \ldots, x_n) = f(x_1, x_1, x_1, x_4, \ldots, x_n)$ or $f(x_1, x_1, x_3, x_4, \ldots, x_n) = f(x_3, x_1, x_3, x_4, \ldots, x_n)$, assuming only that f is monotone in its first three variables.

63. $\langle xyz \rangle = \langle xxyyz \rangle$. *Note:* Emil Post proved, in fact, that a single subroutine for *any* nontrivial monotone self-dual function will suffice to compute them all. (By induction on n, at least one appropriate way to call such an n-ary subroutine will yield $\langle xyz \rangle$.)

64. [*FOCS* **3** (1962), 149–157.] (a) If f is monotone and self-dual, Theorem P says that $f(x) = x_k$ or $f(x) = \langle f_1(x) f_2(x) f_3(x) \rangle$. The condition therefore holds either immediately or by induction. Conversely, if the condition holds it implies that f is monotone (when x and y differ in just one bit) and self-dual (when they differ in all bits).

(b) We merely need to show that it is possible to define f at one new point without introducing a conflict. Let x be the lexicographically smallest point where $f(x)$ is undefined. If $f(\bar{x})$ is defined, set $f(x) = \overline{f(\bar{x})}$. Otherwise if $f(x') = 1$ for some $x' \subseteq x$, set $f(x) = 1$; otherwise set $f(x) = 0$. Then the condition still holds.

65. If $\mathcal{F}$ is maximal intersecting, we have (i) $X \in \mathcal{F} \implies \overline{X} \notin \mathcal{F}$, where $\overline{X}$ is the complementary set $\{1, 2, \ldots, n\} \setminus X$; (ii) $X \in \mathcal{F}$ and $X \subseteq Y \implies Y \in \mathcal{F}$, because $\mathcal{F} \cup \{Y\}$ is intersecting; and (iii) $X \notin \mathcal{F} \implies \overline{X} \in \mathcal{F}$, because $\mathcal{F} \cup \{X\}$ must contain an element $Y \subseteq \overline{X}$. Conversely, one can prove without difficulty that any family $\mathcal{F}$ satisfying (i) and (ii) is intersecting, and maximal if it also satisfies (iii).

Punch line: All three statements are simple, in the language of Boolean functions: (i) $f(x) = 1 \implies f(\bar{x}) = 0$; (ii) $x \subseteq y \implies f(x) \le f(y)$; (iii) $f(x) = 0 \implies f(\bar{x}) = 1$.

66. [T. Ibaraki and T. Kameda, *IEEE Transactions on Parallel and Distributed Systems* **4** (1993), 779–794.] Every family with the property that $Q \subseteq Q'$ implies $Q = Q'$

clearly corresponds to the prime implicants of a monotone Boolean function f. The further condition that $Q \cap Q' \neq \emptyset$ corresponds to the further relation $f(\bar{x}) \leq \overline{f(x)}$, because $f(\bar{x}) = f(x) = 1$ holds if and only if x and $\bar{x}$ both make prime implicants true.

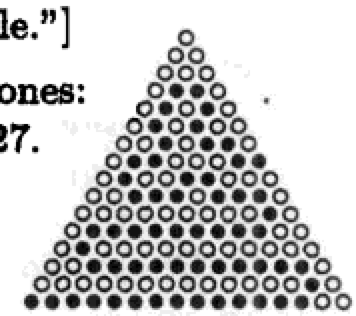
If coteries C and C' correspond in this way to functions f and f', then C dominates C' if and only if $f \neq f'$ and $f'(x) \leq f(x)$ for all x. Then f' is not self-dual, because there is an x with $f'(\bar{x}) = 0$, $f(\bar{x}) = 1$; and we have $f(x) = 0$, hence $f'(x) = 0$.

Conversely, if f' is not self-dual, there's a y with $f'(y) = f'(\bar{y}) = 0$. If $y = 0\ldots0$, coterie C' is empty, and dominated by every other coterie. Otherwise define $f(x) = f'(x) \vee [x \supseteq y]$. Then f is monotone, and $f(\bar{x}) \leq \overline{f(x)}$ for all x; so it corresponds to a coterie that dominates C'.

67. (a) By induction, if $Y(t_{100}) = Y(t_{010}) = 1$ we have a Y of 1s, because the Ys in t_{100} and t_{010} either intersect or are adjacent. Similarly, if $Y(t_{100}) = Y(t_{010}) = 0$ we do not have a Y of 1s, because there is a Y of 0s.

(b) This formula follows from (a) and the fact that $(t_{abc})_{def} = t_{(a+d)(b+e)(c+f)} = (t_{def})_{abc}$. [Schensted stated the results of this exercise, and those of exercises 62 and 69, in a 28-page letter sent to Martin Gardner on 21 January 1979. Milnor had written to Gardner on 26 March 1957 about a corresponding game called "Triangle."]

68. When $n = 15$, the author's best attempt so far has 59 black stones: (The answers for $1 \leq n \leq 10$ are respectively 2, 3, 4, 6, 8, 11, 14, 18, 23, 27. The prime implicants for these functions can be represented by fairly small ZDDs; see Section 7.1.4.)



69. The proof of Theorem P shows that we need only prove $Y(T) \leq f(x)$. A Y in T means that we've got at least one variable in each p_j. Therefore $f(\bar{x}_1, \ldots, \bar{x}_n) = 0$, and $f(x_1, \ldots, x_n) = 1$.

70. Self-duality of g is obvious for arbitrary t when f is self-dual: $\overline{g(\bar{x})} = (\overline{f(\bar{x})} \vee [\bar{x} = t]) \wedge [\bar{x} \neq \bar{t}] = (f(x) \vee [x = \bar{t}]) \wedge [x \neq t] = (f(x) \wedge [x \neq t]) \vee ([x = \bar{t}] \wedge [x \neq t]) = g(x)$.

Let $x = x_1 \ldots x_{j-1} 0 x_{j+1} \ldots x_n$ and $y = x_1 \ldots x_{j-1} 1 x_{j+1} \ldots x_n$; for monotonicity we must prove that $g(x) \leq g(y)$. If $x = t$ or $y = t$, we have $g(x) = 0$; if $x = \bar{t}$ or $y = \bar{t}$, we have $g(y) = 1$; otherwise $g(x) = f(x) \leq f(y) = g(y)$. [*European J. Combinatorics* **16** (1995), 491–501; discovered independently by J. C. Bioch and T. Ibaraki, *IEEE Transactions on Parallel and Distributed Systems* **6** (1995), 905–914.]

71. $\langle\langle xyz\rangle uv\rangle = \langle\langle\langle xyz\rangle uv\rangle uv\rangle = \langle\langle\langle yuv\rangle x\langle zuv\rangle\rangle uv\rangle = \langle\langle yuv\rangle\langle xuv\rangle\langle\langle zuv\rangle uv\rangle\rangle = \langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle$.

72. For (58), $v = \langle uvu\rangle = u$. For (59), $\langle uyv\rangle = \langle vu\langle xuy\rangle\rangle = \langle\langle vux\rangle uy\rangle = \langle xuy\rangle = y$. And for (60), $\langle xyz\rangle = \langle\langle xuv\rangle yz\rangle = \langle x\langle uyz\rangle\langle vyz\rangle\rangle = \langle xyy\rangle = y$.

73. (a) If $d(u,v) = d(u,x) + d(x,v)$, we obviously obtain a shortest path of the form $u — \cdots — x — \cdots — v$. Conversely, if $[uxv]$, let $u — \cdots — x — \cdots — v$ be a shortest path, with l steps to x followed by m steps to v. Then $d(u,v) = l + m \geq d(u,x) + d(x,v) \geq d(u,v)$.

(b) For all z, $\langle zxu\rangle = \langle z\langle vux\rangle\langle yux\rangle\rangle = \langle\langle zvy\rangle ux\rangle \in \{\langle yux\rangle, \langle vux\rangle\} = \{u, x\}$.

(c) We can assume that $d(x,u) \geq d(x,v) > 0$. Let $u — \cdots — y — v$ be a shortest path, and let $w = \langle xuy\rangle$. Then $\langle vxw\rangle = \langle v\langle vux\rangle\langle wux\rangle\rangle = \langle\langle vvw\rangle ux\rangle = \langle vux\rangle = x$, so $x \in [w\,..\,v]$. We have $[uwy]$, because $d(u,y) < d(u,v)$ and $w \in [u\,..\,y]$. If $w \neq u$ we have $d(w,v) < d(u,v)$; hence $[wxv]$, hence $[uxv]$. If $w = u$ we have $x — u$ by (b). But $d(x,u) \geq d(x,v)$; therefore $x — v$, and $[uxv]$.

(d) Let $y = \langle uxv \rangle$. Since $y \in [u \mathinner{..} x]$, we have $d(u,x) = d(u,y) + d(y,x)$ by (a) and (c). Similarly, $d(u,v) = d(u,y) + d(y,v)$ and $d(x,v) = d(x,y) + d(y,v)$. But these three equations, together with $d(u,v) = d(u,x) + d(x,v)$, yield $d(x,y) = 0$. [*Proc. Amer. Math. Soc.* **12** (1961), 407–414.]

74. $w = \langle yxw \rangle = \langle yx\langle zxw \rangle\rangle = \langle yx\langle zx\langle yzw \rangle\rangle\rangle = \langle\langle yxz \rangle x\langle yzw \rangle\rangle = \langle x\langle xyz \rangle\langle wyz \rangle\rangle = \langle\langle xxw \rangle yz \rangle = \langle xyz \rangle$ by (55), (55), (55), (52), (51), (53), and (50).

75. (a) If $w = \langle xxy \rangle$ we have $[xwx]$ by (iii), hence $w = x$ by (i).

(b) Axiom (iii) and part (a) tell us that $[xxy]$ is always true. So we can set $x = y$ in (ii) to conclude that $[uxv] \iff [vxu]$. The definition of $\langle xyz \rangle$ in (iii) is therefore perfectly symmetrical between x, y, and z.

(c) By the definition of $\langle uxv \rangle$ in (iii), we have $x = \langle uxv \rangle$ if and only if $[uxx]$, $[uxv]$, and $[xxv]$. But we know that $[uxx]$ and $[xxv]$ are always true.

(d) In this step and subsequent steps, we will construct one or more auxiliary points of M and then use Algorithm C to derive every consequence of the betweenness relations that are known. (The axioms have the convenient form of Horn clauses.) For example, here we define $z = \langle xyv \rangle$, so that we know $[uxy]$, $[uyv]$, $[xzy]$, $[xzv]$, and $[yzv]$. From these hypotheses we deduce $[uzy]$ and $[uzv]$. So $z = \langle uyv \rangle = y$.

(e) The hinted construction implies, among other things, $[utv]$, $[utz]$, $[vtz]$, $[uwv]$, $[uwz]$, $[vwz]$; hence $t = w$. (A computer program is helpful here.) Adding the hypotheses $[rws]$, $[rwz]$, $[swz]$ now yields $[xyz]$ as desired; it also turns out that $r = p$ and $s = q$.

(f) Let $r = \langle yuv \rangle$, $s = \langle zuv \rangle$, $t = \langle xyz \rangle$, $p = \langle xrs \rangle$, $q = \langle tuv \rangle$; then $[pqp]$ flows out. [*Proc. Amer. Math. Soc.* **5** (1954), 801–807. For early studies of betweenness axioms, see E. V. Huntington and J. R. Kline, *Trans. Amer. Math. Soc.* **18** (1917), 301–325.]

76. Axiom (i) obviously holds, and axiom (ii) follows from commutativity and (52). The answer to exercise 74 derives (iii) from the identity $\langle xyz \rangle = \langle x\langle xyz \rangle\langle wyz \rangle\rangle$; so we need only verify that formula: $\langle x\langle xyz \rangle\langle wyz \rangle\rangle = \langle\langle yxz \rangle x\langle wyz \rangle\rangle = \langle\langle\langle yxz \rangle xz \rangle x\langle wyz \rangle\rangle = \langle\langle yxz \rangle x\langle zx\langle wyz \rangle\rangle\rangle = \langle x\langle xyz \rangle\langle z\langle xyz \rangle w \rangle\rangle = \langle\langle x\langle xyz \rangle z \rangle\langle xyz \rangle w \rangle = \langle\langle xyz \rangle\langle xyz \rangle w \rangle$.

Notes: The original treatment of median algebra by Birkhoff and Kiss in *Bull. Amer. Math. Soc.* **53** (1947), 749–752, assumed (50), (51), and the short distributive law (53). The fact that associativity (52) actually implies distributivity was not realized until many years later; M. Kolibiar and T. Marcisová, *Matematický Časopis* **24** (1974), 179–185, proved it via Sholander's axioms as in this exercise. A mechanical derivation of (53) from (50)–(52) was found in 2005 by R. Veroff and W. McCune, using an extension of the Otter theorem prover.

77. (a) In coordinate $r \mathbin{—} s$ of the labels, suppose $l(r)$ has a 0 and $l(s)$ has a 1; then the left vertices have 0 in that coordinate. If $u \mathbin{—} v \mathbin{—} u'$, where u and u' are on the left but v is on the right, $\langle uu'v \rangle$ lies on the left. But $[u \mathinner{..} v] \cap [u' \mathinner{..} v] = \{v\}$, unless $u = u'$.

(b) This statement is obvious, by Corollary C.

(c) Suppose $u \mathbin{—} v$ and $u' \mathbin{—} v'$, where u and u' are on the left, v and v' are on the right. Let $v = v_0 \mathbin{—} \cdots \mathbin{—} v_k = v'$ be a shortest path, and let $u_0 = u$, $u_k = u'$. All vertices v_j lie on the right, by (b). The left vertex $u_1 = \langle u_0 v_1 u_k \rangle$ must be a common neighbor of u_0 and v_1, since the distance $d(u_0, v_1) = 2$. (We cannot have $u_1 = u_0$, because that would imply the existence of a shortest path from v to v' going through the left vertex u.) Therefore v_1 has rank 1; and so do v_2, …, v_{k-1}, by the same argument. [L. Nebeský, *Commentationes Mathematicæ Universitatis Carolinæ* **12** (1971), 317–325; M. Mulder, *Discrete Math.* **24** (1978), 197–204.]

(d) These steps visit all vertices v of rank 1 in order of their distance $d(v,s)$ from s. If such a v has a late neighbor u not yet seen, the rank of u must be 1 or 2. If the

rank is 1, u will have at least two early neighbors, namely v and the future MATE(u). Step I8 bases its decision on an arbitrary early neighbor w of u such that $w \neq v$. Since $d(w,v) = 2$, the vertex $x = \langle svw \rangle$ has rank 1 by (c). If w has rank 0, then $x = v$; so u has rank 1. Otherwise $d(x,s) < d(v,s)$, and the rank of w was correctly determined when x was visited. If w has rank 1, u lies on a shortest path from v to w; if w has rank 2, w lies on a shortest path from u to s. In both cases u and w have the same rank, by (c).

(e) The algorithm removes all edges equivalent to r — s, by (a) and (d). Their removal clearly disconnects the graph; the two pieces that remain are convex by (b), so they are connected and in fact they are median graphs. Step I7 records all of the relevant relations between the two pieces, because all 4-cycles that disappear are examined there. By induction on the number of vertices, each piece is properly labeled.

78. Every time v appears in step I4, it loses one of its neighbors u_j. Each of these edges v — u_j corresponds to a different coordinate of the labels, so we can assume that $l(v)$ has the form $\alpha 1^k$ for some binary string α. The labels for u_1, u_2, ..., u_k are then $\alpha 01^{k-1}$, $\alpha 101^{k-2}$, ..., $\alpha 1^{k-1}0$. By taking componentwise medians, we can now prove that all 2^k labels of the form $\alpha\beta$ occur for vertices in the graph, since $\langle(\alpha\beta)(\alpha\beta')(0\ldots0)\rangle$ is the bit string $\alpha(\beta\mathbin{\&}\beta')$.

79. (a) If $l(v) = k$, exactly $\nu(k)$ smaller vertices are neighbors of v.

(b) At most $\lfloor n/2 \rfloor$ 1s appear in bit position j, for $0 \le j < \lceil \lg n \rceil$.

(c) Suppose exactly k vertices have labels beginning with 0. At most $\min(k, n-k)$ edges correspond to that bit position, and at most $f(k) + f(n-k)$ other edges are present. But

$$f(n) = \max_{0 \le k \le n} \bigl(\min(k, n-k) + f(k) + f(n-k)\bigr),$$

because the function $g(m,n) = f(m+n) - m - f(m) - f(n)$ satisfies the recurrence

$$g(2m+a, 2n+b) = ab + g(m+a, n) + g(m, n+b) \qquad \text{for } 0 \le a, b \le 1.$$

It follows by induction that $g(m,m) = g(m,m+1) = 0$, and that $g(m,n) \ge 0$ when $m \le n$. [*Annals of the New York Academy of Sciences* **175** (1970), 170–186; D. E. Knuth, *Proc. IFIP Congress 1971* (1972), 24.]

80. (a) (Solution by W. Imrich.) The graph with vertex labels 0000, 0001, 0010, 0011, 0100, 0110, 0111, 1100, 1101, 1110, 1111 cannot be labeled in any essentially different way; but the distance from 0001 to 1101 is 4, not 2.

(b) The cycle C_{2m} is a partial cube, because its vertices can be labeled $l(k) = 1^k 0^{m-k}$, $l(m+k) = 0^k 1^{m-k}$ for $0 \le k < m$. But the bitwise median of $l(0)$, $l(m-1)$, and $l(m+1)$ is $01^{m-2}0$; and indeed those vertices don't have a median, when $m > 2$.

81. Yes. A median graph is a subgraph of a hypercube, which is bipartite.

82. The general case reduces to the simple case where G has only two vertices $\{0,1\}$, because we can operate componentwise on the median labels, and because $d(u,v)$ is the Hamming distance between $l(u)$ and $l(v)$.

In the simple case, the stated rule sets $u_k \leftarrow v_k$ except when $u_{k-1} = v_{k-1} = v_{k+1} \neq v_k$, and it is readily proved optimum. (Other optimum possibilities do exist, however; for example, if $v_0v_1v_2v_3 = 0110$, we could set $u_0u_1u_2u_3 = 0000$.)

[This problem was motivated by the study of self-organizing data structures. F. R. K. Chung, R. L. Graham, and M. E. Saks, in *Discrete Algorithms and Complexity* (Academic Press, 1987), 351–387, have proved that median graphs are the *only* graphs for which u_k can always be chosen optimally as a function of $(v_0, v_1, \ldots, v_{k+1})$, regardless of the subsequent values $(v_{k+2}, \ldots, v_t)$. They have also characterized all cases for which a given finite amount of lookahead will suffice, in *Combinatorica* **9** (1989), 111–131.]

83. Again it suffices to consider the simple two-vertex case. An optimum solution in that case can never have $u_{k-1} \neq u_k$ except when $u_{k-1} = v_{k-1} \neq v_k = u_k$. Therefore we must only decide which runs of consecutive 0s or 1s in $v_0v_1 \ldots v_t$ should induce matching runs in $u_0u_1 \ldots u_t$.

Suppose $v_{k-1}v_k \ldots v_{k+r} = 01^r0$ and $u_{k-1} = 0$. The cost of $u_{k-1}u_k \ldots u_{k+r} = 00\ldots0$ is $r\rho$, compared to a cost of 2 if $u_{k-1}u_k \ldots u_{k+r} = 01^r0$; so we must choose the former if $\rho < 2/r$ and the latter if $\rho > 2/r$. This policy can be expressed in terms of medians, if we set $u_k = \langle u_{k-1} \ldots u_{k-1}v_k \ldots v_{k+r}\rangle$ when $2/(r+1) \le \rho < 2/r$ for $r \ge 0$, where there are r occurrences of u_{k-1}. (This median-of-$(2r+1)$ gives correct results in a general median graph, since it will produce valid median labels; see exercise 86.)

But we must change this rule when we get near the end, because it eventually asks for the values of v_j with $j > t$. The endgame is tricky; for if $v_{k-1}v_k \ldots v_t = 01^r$ and $u_{k-1} = 0$, with $r = t+1-k$, the choice $u_{k-1}u_k \ldots u_t = 00\ldots0$ is now preferable only if $\rho < 1/r$. For example, if $2/4 < \rho < 2/3$ and $v_0v_1 = 01$ and $t = 1$, we want to choose $u_0u_1 = 00$; but if $v_0v_1v_2 = 011$ and $t = 2$ we want $u_0u_1u_2 = 011$. One solution is to set $v_{t+j} \leftarrow v_t$ for $1 \le j \le \lceil r/2 \rceil$ and $v_{t+j} \leftarrow v_0$ for $\lceil r/2 \rceil < j \le r$.

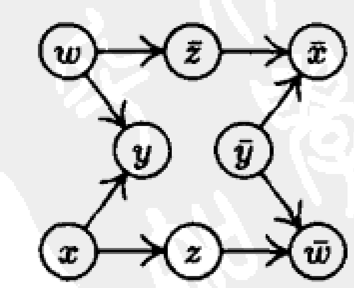
84. There are 81 such functions, each of which can be represented as the median of an odd number of elements. Seven types of vertices occur:

Type	Typical vertex	Cases	Adjacent to	Degree
1	$\langle z\rangle$	5	$\langle vwxyzzz\rangle$	1
2	$\langle vwxyzzz\rangle$	5	$\langle z\rangle, \langle wxyzz\rangle$	5
3	$\langle wxyzz\rangle$	20	$\langle vwxyzzz\rangle, \langle vwxxyyzzz\rangle$	4
4	$\langle vwxxyyzzz\rangle$	30	$\langle xyz\rangle, \langle wxyzz\rangle, \langle vwxyyzz\rangle$	5
5	$\langle vwxyyzz\rangle$	10	$\langle vwxxyyzzz\rangle, \langle vwxyz\rangle$	7
6	$\langle vwxyz\rangle$	1	$\langle vwxyyzz\rangle$	10
7	$\langle xyz\rangle$	10	$\langle vwxxyyzzz\rangle$	3

[Von Neumann and Morgenstern enumerated these seven types in their book *Theory of Games and Economic Behavior* (1944), §52.5, in connection with the study of an equivalent problem about systems of winning coalitions that they called *simple games*. The graph for six-variable functions, which has 2646 vertices of 30 types, appears in the paper by Meyerowitz cited in exercise 70. Only 21 of those types can be represented as a simple median-of-odd; a vertex like $\langle\langle abd\rangle\langle ace\rangle\langle bcf\rangle\rangle$, for example, has no such representation. Let the corresponding graph for n variables have M_n vertices; P. Erdös and N. Hindman, in *Discrete Math.* **48** (1984), 61–65, showed that $\lg M_n$ is asymptotic to $\binom{n-1}{\lfloor n/2 \rfloor}$. D. Kleitman, in *J. Combin. Theory* **1** (1966), 153–155, showed that the vertices for distinct projection functions like x and y are always furthest apart in this graph.]

85. Every strong component must consist of a single vertex; otherwise two coordinates would always be equal, or always complementary. Thus the digraph must be acyclic.

Furthermore, there must be no path from a vertex to its complement; otherwise a coordinate would be constant.

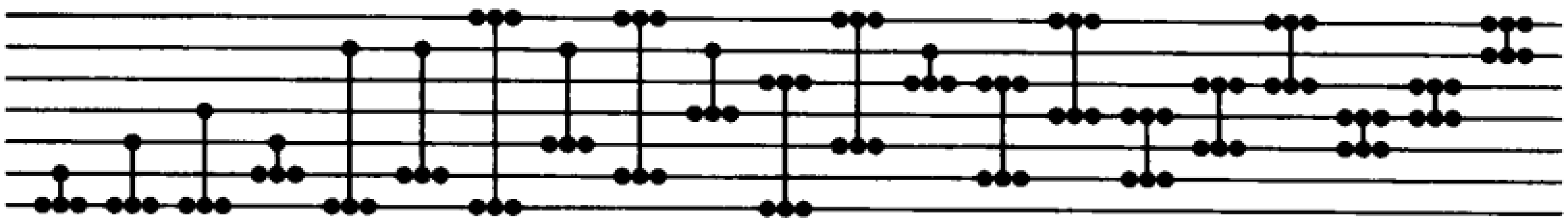


When these two conditions are satisfied, we can prove that no vertex x is redundant, by assigning the value 0 to all vertices that precede x or $\bar{x}$, assigning 1 to all vertices that follow, and giving appropriate values to all other vertices.

(Consequently we obtain a completely different way to represent a median graph. For example, the digraph shown corresponds to the median graph whose labels are $\{0000, 0001, 0010, 0011, 0111, 1010\}$.)

86. Yes. By Theorem P, *any* monotone self-dual function maps elements of X into X.

87. Here the topological ordering $7\,6\,5\,4\,3\,2\,1\,\bar{1}\,\bar{2}\,\bar{3}\,\bar{4}\,\bar{5}\,\bar{6}\,\bar{7}$ can replace (72); we get

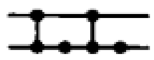


(Consecutive inverters on the same line can, of course, be canceled out.)

88. A given value of d contributes at most $6\lceil t/d\rceil$ units of delay (for $2\lceil t/d\rceil$ clusters).

89. Suppose first that the new condition is $i \to j$ while the old was $i' \to j'$, where $i < j$ and $i' < j'$ and there are no complemented literals. The new module changes $x_1 \ldots x_t$ to $y_1 \ldots y_t$, where $y_i = x_i \wedge x_j$, $y_j = x_i \vee x_j$, and $y_k = x_k$ otherwise. We certainly have $y_{i'} \le y_{j'}$ when $\{i', j'\} \cap \{i, j\} = \emptyset$. And there is no problem if $i = i'$, since $y_{i'} = y_i \le x_i = x_{i'} \le x_{j'} = y_{j'}$. But the case $i = j'$ is trickier: Here the relations $i' \to i$ and $i \to j$ imply also $i' \to j$; and this relation has been enforced by *previous* modules, because modules have been appended in order of decreasing distance d in the topological ordering $u_1 \ldots u_{2t}$. Therefore $y_{i'} = x_{i'} \le x_j$ and $y_{i'} \le x_{j'} = x_i$, hence $y_{i'} \le x_i \wedge x_j = y_i = y_{j'}$. A similar proof works when $j = i'$ or $j = j'$.

Finally, with complemented literals, the construction cleverly reduces the general case to the uncomplemented case by inverting and un-inverting the bits.

90. When $t = 2$,  does the job. The general case follows recursively from this building block by reducing t to $\lceil t/2\rceil$.

[The study of CI-nets, and other networks of greater generality, was initiated by E. W. Mayr and A. Subramanian, *J. Computer and System Sci.* **44** (1992), 302–323.]

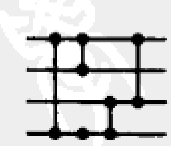
91. The answer does not yet seem to be known even in the special case when the median graph is a free tree (with $t + 1$ vertices), or in the monotone case when it is a distributive lattice as in Corollary F. In the latter case, inverters may be unnecessary.

93. Let $d_X(u, v)$ be the number of edges on a shortest path between u and v, when the path lies entirely within X. Clearly $d_X(u, v) \ge d_G(u, v)$. And if $u = u_0 — u_1 — \cdots — u_k = v$ is a shortest path in G, the path $u = f(u_0) — f(u_1) — \cdots — f(u_k) = v$ lies in X when f is a retraction from G to X; hence $d_X(u, v) \le d_G(u, v)$.

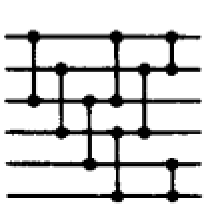
94. If f is a retraction of the t-cube onto X, two different coordinate positions cannot always be equal or always complementary for all $x \in X$, unless they are constant. For if, say, all elements of X have the forms $00{*}\ldots{*}$ or $11{*}\ldots{*}$, there would be no path between vertices of those two types, contradicting the fact that X is an isometric subgraph (hence connected).

Given $x, y, z \in X$, let $w = \langle xyz\rangle$ be their median in the t-cube. Then $f(w) \in [x\,..\,y] \cap [x\,..\,z] \cap [y\,..\,z]$, because (for example) $f(w)$ lies on a shortest path from x to y in X. So $f(w) = w$, and we have proved that $w \in X$. [This result and its considerably more subtle converse are due to H. J. Bandelt, *J. Graph Theory* **8** (1984), 501–510.]

95. False (although the author was hoping otherwise); the network at the right takes $0001 \mapsto 0000$, $0010 \mapsto 0011$, $1101 \mapsto 0110$, but nothing $\mapsto 0010$.



(The set of all possible outputs appears to have no easy characterization, even when no inverters are used. For example, the pure-comparator network at the left, constructed by Tomás Feder, takes $000000 \mapsto 000000$, $010101 \mapsto 010101$, and $101010 \mapsto 011001$, but nothing $\mapsto 010001$. See also exercises 5.3.4–50, 5.3.4–52.)



96. No. If f is a threshold function based on real parameters $w = (w_1, \ldots, w_n)$ and t, let $\max\{w \cdot x \mid f(x) = 0\} = t - \epsilon$. Then $\epsilon > 0$, and f is defined by the 2^n inequalities $w \cdot x - t \geq 0$ when $f(x) = 1$, $t - w \cdot x - \epsilon \geq 0$ when $f(x) = 0$. If A is any $M \times N$ matrix of integers for which the system of linear inequalities $Av \geq (0, \ldots, 0)^T$ has a real-valued solution $v = (v_1, \ldots, v_N)^T$ with $v_N > 0$, there also is such a solution in integers. (Proof by induction on N.) So we can assume that w_1, …, w_n, t, and ϵ are integers.

[A closer analysis using Hadamard's inequality (see Eq. 4.6.1–(25)) proves in fact that integer weights of magnitude at most $(n+1)^{(n+1)/2}/2^n$ will suffice; see S. Muroga, I. Toda, and S. Takasu, *J. Franklin Inst.* **271** (1961), 376–418, Theorem 16. Furthermore, exercise 112 shows that weights nearly that large are sometimes needed.]

97. $\langle 11111x_1x_2\rangle$, $\langle 111x_1x_2\rangle$, $\langle 1x_1x_2\rangle$, $\langle 0x_1x_2\rangle$, $\langle 000x_1x_2\rangle$, $\langle 00000x_1x_2\rangle$.

98. We may assume that $f(x_1, \ldots, x_n) = \langle y_1^{w_1} \ldots y_n^{w_n}\rangle$, with positive integer weights w_j and with $w_1 + \cdots + w_n$ odd. Let δ be the minimum positive value of the 2^n sums $\pm w_1 \pm \cdots \pm w_n$, with n independently varying signs. Renumber all subscripts so that $w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n = \delta$. Then $w_1y_1 + \cdots + w_ny_n > \frac{1}{2}(w_1 + \cdots + w_n) \iff w_1(y_1 - \frac{1}{2}) + \cdots + w_n(y_n - \frac{1}{2}) > 0 \iff w_1(y_1 - \frac{1}{2}) + \cdots + w_n(y_n - \frac{1}{2}) > -\delta/2 \iff w_1y_1 + \cdots + w_ny_n > \frac{1}{2}(w_1 + \cdots + w_n - (w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n)) = w_{k+1} + \cdots + w_n \iff w_1y_1 + \cdots + w_ky_k - w_{k+1}\bar{y}_{k+1} - \cdots - w_n\bar{y}_n > 0$.

99. We have $[x_1 + \cdots + x_{2s-1} + s(y_1 + \cdots + y_{2t-2}) \geq st] = [\lfloor (x_1 + \cdots + x_{2s-1})/s\rfloor + y_1 + \cdots + y_{2t-2} \geq t]$; and $\lfloor (x_1 + \cdots + x_{2s-1})/s\rfloor = [x_1 + \cdots + x_{2s-1} \geq s]$.

(For example, $\langle\langle xyz\rangle uv\rangle = \langle xyzu^2v^2\rangle$, a quantity that we also know is equal to $\langle x\langle yuv\rangle\langle zuv\rangle\rangle$ and $\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle$ by Eqs. (53) and (54). *Reference:* C. C. Elgot, *FOCS* **2** (1961), 238.)

100. True, because of the preceding exercise and (45).

101. (a) When $n = 7$ they are $x_7 \wedge x_6$, $x_6 \wedge x_5$, $x_7 \wedge x_5 \wedge x_4$, $x_6 \wedge x_4 \wedge x_3$, $x_7 \wedge x_5 \wedge x_3 \wedge x_2$, $x_6 \wedge x_4 \wedge x_2 \wedge x_1$, $x_7 \wedge x_5 \wedge x_3 \wedge x_1$; and in general there are n prime implicants, forming a similar pattern. (We have either $x_n = x_{n-1}$ or $x_n = \bar{x}_{n-1}$. In the first case, $x_n \wedge x_{n-1}$ is obviously a prime implicant. In the second case, $F_n(x_1, \ldots, x_{n-1}, \bar{x}_{n-1}) = F_{n-1}(x_1, \ldots, x_{n-1})$; so we use the prime implicants of the latter, and insert x_n when x_{n-1} does not appear.)

(b) The shelling pattern (0000011, $000011\underline{0}$, $00011\underline{0}1$, $0011\underline{0}1\underline{0}$, $011\underline{0}1\underline{0}1$, $11\underline{0}1\underline{0}1\underline{0}$, $1\underline{0}1\underline{0}1\underline{0}1$) for $n = 7$ works for all n.

(c) Two of several possibilities for $n = 7$ illustrate the general case:

$$F_7(x_1, \ldots, x_7) = Y\left(\begin{array}{c} x_6 \\ x_7\ x_5 \\ x_6\ x_6\ x_4 \\ x_7\ x_5\ x_7\ x_3 \\ x_6\ x_6\ x_4\ x_6\ x_2 \\ x_7\ x_5\ x_7\ x_3\ x_7\ x_1 \end{array}\right) = Y\left(\begin{array}{c} x_6 \\ x_7\ x_5 \\ x_6\ x_6\ x_4 \\ x_7\ x_5\ x_5\ x_3 \\ x_6\ x_6\ x_4\ x_4\ x_2 \\ x_7\ x_5\ x_5\ x_3\ x_3\ x_1 \end{array}\right).$$

[The Fibonacci threshold functions were introduced by S. Muroga, who also discovered the optimality result in exercise 105; see *IEEE Transactions* **EC-14** (1965), 136–148.]

102. (a) By (1) and (2), $\hat{f}(\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_n)$ is the complement of $\hat{f}(x_0, x_1, \ldots, x_n)$.

(b) If f is given by (75), $\hat{f}$ is $[(w + 1 - 2t)x_0 + w_1x_1 + \cdots + w_nx_n \geq w + 1 - t]$, where $w = w_1 + \cdots + w_n$. Conversely, if $\hat{f}$ is a threshold function, so is $f(x_1, \ldots, x_n) = \hat{f}(1, x_1, \ldots, x_n)$. [E. Goto and H. Takahasi, *Proc. IFIP Congress* (1962), 747–752.]

103. [See R. C. Minnick, *IRE Transactions* **EC-10** (1961), 6–16.] We want to minimize $w_1 + \cdots + w_n$ subject to the constraints $w_j \geq 0$ for $1 \leq j \leq n$ and $(2e_1 - 1)w_1 + \cdots + (2e_n - 1)w_n \geq 1$ for each prime implicant $x_1^{e_1} \wedge \cdots \wedge x_n^{e_n}$. For example, if $n = 6$, the prime implicant $x_2 \wedge x_5 \wedge x_6$ would lead to the constraint $-w_1 + w_2 - w_3 - w_4 + w_5 + w_6 \geq 1$. If the minimum is $+\infty$, the given function is not a threshold function. (The answer to exercise 84 gives one of the simplest examples of such a case.) Otherwise, if the solution $(w_1, \ldots, w_n)$ involves only integers, it minimizes the desired size. When noninteger solutions arise, additional constraints must be added until the best solution is found, as in part (c) of the following exercise.

104. First we need an algorithm to generate the prime implicants $x_1^{e_1} \wedge \cdots \wedge x_n^{e_n}$ of a given majority function $\langle x_1^{w_1} \ldots x_n^{w_n} \rangle$, when $w_1 \geq \cdots \geq w_n$ and $w_1 + \cdots + w_n$ is odd:

K1. [Initialize.] Set $t \leftarrow 0$. Then for $j = n, n-1, \ldots, 1$ (in this order), set $a_j \leftarrow t$, $t \leftarrow t + w_j$, $e_j \leftarrow 0$. Finally set $t \leftarrow (t+1)/2$, $s_1 \leftarrow 0$, and $l \leftarrow 0$.

K2. [Enter level l.] Set $l \leftarrow l + 1$, $e_l \leftarrow 1$, $s_{l+1} \leftarrow s_l + w_l$.

K3. [Below threshold?] If $s_{l+1} < t$, return to K2.

K4. [Visit a prime implicant.] Visit the exponents $(e_1, \ldots, e_n)$.

K5. [Downsize.] Set $e_l \leftarrow 0$. Then if $s_l + a_l \geq t$, set $s_{l+1} \leftarrow s_l$ and go to K2.

K6. [Backtrack.] Set $l \leftarrow l - 1$. Terminate if $l = 0$; otherwise go to K5 if $e_l = 1$; otherwise repeat this step. ▌

(a) $\langle x_1 x_2^2 x_3^3 x_4^5 x_5^6 x_6^8 x_7^{10} x_8^{12} \rangle$ (21 prime implicants).

(b) The optimum weights for $\langle x_0^{16-2t} x_1^8 x_2^4 x_3^2 x_4 \rangle$ are $w_0 w_1 w_2 w_3 w_4 = 10000$, 31111, 21110, 32211, 11100, 23211, 12110, 13111, 01000, for $0 \leq t \leq 8$; the other cases are dual.

(c) Here the optimum weights $(w_1, \ldots, w_{10})$ are $(29, 25, 19, 15, 12, 8, 8, 3, 3, 0)/2$; so we learn that x_{10} is irrelevant, and we must deal with fractional weights. Constraining $w_8 \geq 2$ gives integer weights $(15, 13, 10, 8, 6, 4, 4, 2, 1, 0)$, which must be optimum because their sum exceeds the previous sum by 2. (Only two of the 175,428 self-dual threshold functions on nine variables have nonintegral weights minimizing $w_1 + \cdots + w_n$; the other one is $\langle x_1^{17} x_2^{15} x_3^{11} x_4^9 x_5^7 x_6^5 x_7^4 x_8^2 x_9 \rangle$. The largest w_1 in a minimum representation occurs in $\langle x_1^{42} x_2^{22} x_3^{18} x_4^{15} x_5^{13} x_6^{10} x_7^8 x_8^4 x_9^3 \rangle$; the largest $w_1 + \cdots + w_9$ occurs uniquely in $\langle x_1^{34} x_2^{32} x_3^{28} x_4^{27} x_5^{24} x_6^{20} x_7^{18} x_8^{15} x_9^{11} \rangle$, which is also an example of the largest w_9. See S. Muroga, T. Tsuboi, and C. R. Baugh, *IEEE Transactions* **C-19** (1970), 818–825.)

105. When $n = 7$, the inequalities generated in exercise 103 are $w_7 + w_6 - w_5 - w_4 - w_3 - w_2 - w_1 \geq 1$, $-w_7 + w_6 + w_5 - w_4 - w_3 - w_2 - w_1 \geq 1$, $w_7 - w_6 + w_5 + w_4 - w_3 - w_2 - w_1 \geq 1$, $-w_7 + w_6 - w_5 + w_4 + w_3 - w_2 - w_1 \geq 1$, $w_7 - w_6 + w_5 - w_4 + w_3 + w_2 - w_1 \geq 1$, $-w_7 + w_6 - w_5 + w_4 - w_3 + w_2 + w_1 \geq 1$, $w_7 - w_6 + w_5 - w_4 + w_3 - w_2 + w_1 \geq 1$. Multiply them respectively by 1, 1, 2, 3, 5, 8, 5 to get $w_1 + \cdots + w_7 \geq 1 + 1 + 2 + 3 + 5 + 8 + 5$. The same idea works for all $n \geq 3$.

106. (a) $\langle x_1^{2^{n-1}} x_2^{2^{n-2}} \ldots x_n \, \bar{y}_1^{2^{n-1}} \bar{y}_2^{2^{n-2}} \ldots \bar{y}_n \bar{z} \rangle$. (By exercise 99, we could also perform n medians-of-three: $\langle \langle \ldots \langle x_n \bar{y}_n \bar{z} \rangle \ldots x_2 \bar{y}_2 \rangle x_1 \bar{y}_1 \rangle$.)

(b) If $\langle x_1^{u_1} x_2^{u_2} \ldots x_n^{u_n} \bar{y}_1^{v_1} \bar{y}_2^{v_2} \ldots \bar{y}_n^{v_n} \bar{z}^w \rangle$ solves the problem, $2^{n+1} - 1$ basic inequalities need to hold; for example, when $n = 2$ they are $u_1 + u_2 - v_1 + v_2 - w \geq 1$, $u_1 + u_2 - v_1 - v_2 + w \geq 1$, $u_1 - u_2 + v_1 - v_2 - w \geq 1$, $u_1 - u_2 - v_1 + v_2 + w \geq 1$, $-u_1 + u_2 + v_1 + v_2 - w \geq 1$, $-u_1 + u_2 + v_1 - v_2 + w \geq 1$, $-u_1 - u_2 + v_1 + v_2 + w \geq 1$. Add them all up to get $u_1 + u_2 + \cdots + u_n + v_1 + v_2 + \cdots + v_n + w \geq 2^{n+1} - 1$.

107.

f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$
$\bot$	0	$(0,0)$	$\bar\subset$	1	$(0,1)$	$\bar\vee$	1	$(0,0)$	$\bar{\mathsf{L}}$	2	$(0,1)$
$\wedge$	1	$(1,1)$	R	2	$(1,2)$	$\equiv$	2	$(1,1)$	$\supset$	3	$(1,2)$
$\bar\supset$	1	$(1,0)$	$\oplus$	2	$(1,1)$	$\bar{\mathsf{R}}$	2	$(1,0)$	$\bar\wedge$	3	$(1,1)$
L	2	$(2,1)$	$\vee$	3	$(2,2)$	$\subset$	3	$(2,1)$	$\top$	4	$(2,2)$

Notice that $\oplus$ and $\equiv$ have the same parameters $N(f)$ and $\Sigma(f)$; they are the only Boolean binary operations that aren't threshold functions.

108. If $\Sigma(g) = (s_0, s_1, \ldots, s_n)$, the value of g is 1 in s_0 cases when $x_0 = 1$ and in $2^n - s_0$ cases when $x_0 = 0$. We also have $\Sigma(f_0) + \Sigma(f_1) = (s_1, \ldots, s_n)$, and

$$\begin{aligned}\Sigma(f_0) &= \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} (\bar{x}_1, \ldots, \bar{x}_n) g(0, \bar{x}_1, \ldots, \bar{x}_n) \\ &= \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} \big((1, \ldots, 1) - (x_1, \ldots, x_n)\big)\big(1 - g(1, x_1, \ldots, x_n)\big) \\ &= (2^{n-1} - s_0, \ldots, 2^{n-1} - s_0) + \Sigma(f_1).\end{aligned}$$

So the answers, for $n > 0$, are (a) $N(f_0) = 2^n - s_0$, $\Sigma(f_0) = \frac{1}{2}(s_1 - s_0 + 2^{n-1}, \ldots, s_n - s_0 + 2^{n-1})$; (b) $N(f_1) = s_0$, $\Sigma(f_1) = \frac{1}{2}(s_1 + s_0 - 2^{n-1}, \ldots, s_n + s_0 - 2^{n-1})$. [Equivalent results were presented by E. Goto in lectures at MIT in 1963.]

109. (a) $a_1 + \cdots + a_k \ge b_1 + \cdots + b_k$ if and only if $k - a_1 - \cdots - a_k \le k - b_1 - \cdots - b_k$.

(b) Let $\alpha^+ = (a_1, a_1{+}a_2, \ldots, a_1{+}\cdots{+}a_n)$. Then the vector $(c_1, \ldots, c_n)$ obtained by componentwise minimization of α^+ and β^+ is $(\alpha \wedge \beta)^+$. (Clearly $c_j = c_{j-1} + a_j$ or b_j.)

(c) Proceed as in (b) but with componentwise *maximization*; or take $\overline{\bar\alpha \wedge \bar\beta}$.

(d) True, because max and min satisfy these distributive laws. (In fact, we obtain a distributive *mixed-radix majorization lattice* in a similar way from the set of all n-tuples $a_1 \ldots a_n$ such that $0 \le a_j < m_j$ for $1 \le j \le n$.)

(e) $\alpha 1$ covers $\alpha 0$ and $\alpha 10\beta$ covers $\alpha 01\beta$. [This characterization is due to R. O. Winder, *IEEE Trans.* **EC-14** (1965), 315–325, but he didn't prove the lattice property. The lattice is often called $M(n)$; see B. Lindström, *Nordisk Mat. Tidskrift* **17** (1969), 61–70; R. P. Stanley, *SIAM J. Algebraic and Discrete Methods* **1** (1980), 177–179.]

(f) Because of (e) we have $r(\alpha) = na_1 + (n-1)a_2 + \cdots + a_n$.

(g) The point is that $0\beta \succeq 0\alpha$ if and only if $\beta \succeq \alpha$ and that $1\beta \succeq 0\alpha$ if and only if $1\beta \succeq 10\ldots0 \vee 0\alpha = 1\alpha'$.

(h) That is, how many $a_1 \ldots a_n$ have the property that $a_1 \ldots a_k$ contains no more 1s than 0s? The answer is $\binom{n}{\lfloor n/2 \rfloor}$; see, for example, exercise 2.2.1–4 or 7.2.1.6–42(a).

110. (a) If $x \subseteq y$ then $x \preceq y$, hence $f(x) \le f(y)$; QED.

(b) No; a threshold function need not be monotone (see (79)). But we *can* show that f is regular if we also require $w_n \ge 0$: For if $f(x) = 1$ and y covers x we then have $w \cdot y \ge w \cdot x$.

(c) Whenever $f(x) = 1$ and $x_j < x_{j+1}$, we have $f(y) = 1$ when y covers x with $x_j \leftrightarrow x_{j+1}$; hence $s_j \ge s_{j+1}$. (This argument holds even when $w_n < 0$.)

(d) No; consider, for example, $\langle x_1 x_2^2 x_3^2 \rangle$, which equals $\langle x_1 x_2 x_3 \rangle$. Counterexamples can arise even when the weights minimize $w_1 + \cdots + w_n$, because the solution to the linear program in exercise 103 is not always unique. One such case, found by Muroga, Tsuboi, and Baugh, is $\langle x_1^{17} x_2^9 x_3^8 x_4^6 x_5^7 x_6^5 x_7^3 x_8^2 x_9^2 \rangle$, a function that is actually symmetric in x_4 and x_5. But if $s_j > s_{j+1}$ we must have $w_j > w_{j+1}$, because of (c).

111. (a) Find an optimum self-dual function f pointwise as in exercise 14; in case of ties, set $f(x_1,\ldots,x_n)=x_1$. Then $f(x_1,\ldots,x_n)=[r_1^{x_1}\ldots r_n^{x_n}\ge\sqrt{r_1\ldots r_n}\,]$, except that '$\ge$' becomes '$>$' when $x_1=0$. This function is regular when $r_1\ge\cdots\ge r_n\ge 1$.

(b) Let g be the regular, self-dual function constructed in (a). If f is a given regular, self-dual function, we want to verify that $f(x)\le g(x)$ for all vectors x; this will imply that $f=g$, because both functions are self-dual.

Suppose $f(x)=1$, and let $y\preceq x$ be minimal such that $f(y)=1$. If we have verified that $g(y)=1$, then indeed $g(x)=1$, as desired. [See K. Makino and T. Kameda, *SIAM Journal on Discrete Mathematics* **14** (2001), 381–407.]

For example, there are only seven self-dual regular Boolean functions when $n=5$, generated by the following minimal elements in Fig. 8: 10000; 01111, 10001; 01110, 10010; 01101, 10011, 10100; 01100; 01011, 11000; 00111. So an optimum coterie can be found by examining only a few function values.

(c) Suppose $1>p_1\ge\cdots\ge p_r\ge\frac12>p_{r+1}\ge\cdots\ge p_n>0$. Let $f_k(x_1,\ldots,x_n)$ be the kth monotone, self-dual function and $F_k(x_1,\ldots,x_n)$ its integer multilinear representation. We want to find the optimum availability $G(p_1,\ldots,p_n)=\max_k F_k(p_1,\ldots,p_n)$. If $p_1\le p_1'$, $\ldots$, $p_n\le p_n'$, we have $F_k(p_1,\ldots,p_n)\le F_k(p_1',\ldots,p_n')$ by exercise 12(e); hence $G(p_1,\ldots,p_n)\le G(p_1',\ldots,p_n')$.

Therefore if $0<r<n$ we have $G(p_1,\ldots,p_n)\le G(p_1,\ldots,p_r,\frac12,\ldots,\frac12)$. And the latter is $F(p_1,\ldots,p_r,\frac12,\ldots,\frac12)$, where f is the function derived in part (a). This function does not depend on $(x_{r+1},\ldots,x_n)$, so it gives the optimum.

If $r=0$ the problem seems to be deeper. We have $G(p_1,\ldots,p_n)\le G(p_1,\ldots,p_1)$; so we can conclude that the optimum coterie is $f(x_1,\ldots,x_n)=x_1$ if we can show that $F_k(p,\ldots,p)\le p$ for all k whenever $p<\frac12$. In general $F_k(p\ldots,p)=\sum_m c_m p^m(1-p)^{n-m}$, where c_m is the number of vectors x such that $f_k(x)=1$ and $\nu x=m$. Exercise 70 tells us that $c_m+c_{n-m}=\binom{n}{m}$, for all k. And the Erdős–Ko–Rado theorem (exercise 7.2.1.3–111) tells us that $c_m\le\binom{n-1}{m-1}$ for any intersecting family of m-sets when $m\le n/2$. The result follows.

[See Y. Amir and A. Wool, *Information Processing Letters* **65** (1998), 223–228.]

112. (a) The leading terms are respectively 0, $+xy$, $-xy$, $+x$, $-xy$, $+y$, $-2xy$, $-xy$, $+xy$, $+2xy$, $-y$, $+xy$, $-x$, $+xy$, $-xy$, 1; so $F(f)=1$ when f is $\wedge$, $\llcorner$, R, $\bar\vee$, $\equiv$, $\subset$, $\supset$, $\top$.

(b) The coefficient corresponding to exponents 01101, say, is $f_{0{*}{*}0{*}}$ in the notation of answer 12; it is a linear combination of truth table entries, always lying in the range $\lceil -2^{k-1}\rceil\le f_{0{*}{*}0{*}}\le\lceil 2^{k-1}\rceil$ when there are k asterisks. Thus the leading coefficient is positive if and only if the mixed-radix number

$$\begin{bmatrix} f_{{*}{*}\ldots{*}}, & f_{0{*}\ldots{*}}, & \ldots, & f_{{*}0\ldots0}, & f_{00\ldots0} \\ 2^m{+}1, & 2^{m-1}{+}1, & \ldots, & 2^1{+}1, & 2^0{+}1 \end{bmatrix}$$

is positive, where the f's are arranged in reverse order of Chase's sequence and the radix 2^k+1 corresponds to an f with k asterisks. For example, when $m=2$ we have $F(f)=1$ if and only if the sum $18f_{{*}{*}}+6f_{0{*}}+2f_{{*}0}+f_{00}=18(f_{11}-f_{01}-f_{10}+f_{00})+6(f_{01}-f_{00})+2(f_{10}-f_{00})+f_{00}=18f_{11}-12f_{01}-16f_{10}+11f_{00}$ is positive; so the threshold function can be written $\langle f_{11}^{18}\bar f_{01}^{12}\bar f_{10}^{16}f_{00}^{11}\rangle$.

(In this particular case the much simpler expression $\langle f_{11}f_{11}\bar f_{01}\bar f_{10}f_{00}\rangle$ is actually valid. But part (c) will show that when m is large we can't do a great deal better.)

(c) Suppose $F(f)=[\sum_\alpha v_\alpha(f_\alpha-\frac12)>0]$, where the sum is over all 2^m binary strings α of length m and where each v_α is an integer weight. Define

$$w_\alpha=\sum_\beta(-1)^{\nu(\alpha\dot-\beta)}v_\beta \qquad\text{and}\qquad F_\alpha=\sum_\beta(-1)^{\nu(\alpha\dot-\beta)}f_\beta-2^{m-1}[\alpha=00\ldots0];$$

thus, for example, $w_{01} = -v_{00} + v_{01} - v_{10} + v_{11}$ and $F_{11} = f_{00} - f_{01} - f_{10} + f_{11}$. One can show that $F_{1^k0^l} = 2^l f_{*^k0^l}$, if $F_\alpha = 0$ whenever $\nu(\alpha) > k > 0$; therefore the signs of the transformed truth coefficients F_α determine the sign of the leading coefficient in the multilinear representation. Furthermore, we now have $F(f) = [\sum_\alpha w_\alpha F_\alpha > 0]$.

The general idea of the proof is to choose test functions f from which we can derive properties of the transformed weights w_α. For example, if $f(x_1, \ldots, x_m) = x_1 \oplus \cdots \oplus x_k$, we find $F_\alpha = 0$ for all α except that $F_{1^k0^{m-k}} = (-1)^{k-1}2^{m-1}$. The multilinear representation of $x_1 \oplus \cdots \oplus x_k$ has leading term $\lceil(-2)^{k-1}\rceil x_1 \ldots x_k$; hence we can conclude that $w_{1^k0^{m-k}} > 0$, and in a similar way that $w_\alpha > 0$ for all α. In general if m changes to $m+1$ but f does not depend on x_{m+1}, we have $F_{\alpha 0} = 2F_\alpha$ and $F_{\alpha 1} = 0$.

The test function $x_2 \oplus \cdots \oplus x_m \oplus x_1\bar{x}_2 \ldots \bar{x}_m$ proves that

$$w_{1^m} > (2^{m-1}-1)w_{01^{m-1}} + \sum_{k=1}^{m-1} w_{1^k01^{m-1-k}} + \text{smaller terms},$$

where the smaller terms involve only w_α with $\nu(\alpha) \le m-2$. In particular, $w_{11} > w_{01} + w_{10} + w_{00}$. The test function $x_1 \oplus \cdots \oplus x_{m-1} \oplus \bar{x}_1 \ldots \bar{x}_{m-2}(x_{m-1} \oplus \bar{x}_m)$ proves

$$w_{1^{m-2}01} > (2^{m-2}-1)w_{1^{m-2}10} + \sum_{k=0}^{m-3}(w_{1^k01^{m-3-k}10} + w_{1^k01^{m-3-k}01}) + \text{smaller terms},$$

where the smaller terms this time have $\nu(\alpha) \le m-3$. In particular, $w_{101} > w_{110} + w_{010} + w_{001}$. By permuting subscripts, we obtain similar inequalities leading to

$$w_{\alpha_j} > (2^{\nu(\alpha_j)-1} - 1)w_{\alpha_{j-1}} \qquad \text{for } 0 < j < 2^m,$$

because the w's begin to grow rapidly. But we have $v_\alpha = \sum_\beta (-1)^{\nu(\beta \dot{-} \alpha)} w_\beta / n$; hence $|v_\alpha| = w_{11\ldots1}/n + O(w_{11\ldots1}/n^2)$. [*SIAM J. Discrete Math.* **7** (1994), 484–492. Important generalizations of this result have been obtained by N. Alon and V. H. Vũ, *J. Combinatorial Theory* **A79** (1997), 133–160.]

113. The stated g_3 is $S_{2,3,6,8,9}$ because the stated g_2 is $S_{2,3,4,5,8,9,10,11,12}$.

For the more difficult function $S_{1,3,5,8}$, let $g_1 = [\nu x \ge 6]$; $g_2 = [\nu x \ge 3]$; $g_3 = [\nu x - 5g_1 - 2g_2 \ge 2] = S_{2,4,5,9,10,11,12}$; $g_4 = [2\nu x - 15g_1 - 9g_3 \ge 1] = S_{1,3,5,8}$. [See M. A. Fischler and M. Tannenbaum, *IEEE Transactions* **C-17** (1968), 273–279.]

114. $[4x + 2y + z \in \{3, 6\}] = (\bar{x} \wedge y \wedge z) \vee (x \wedge y \wedge \bar{z})$. In the same way, *any* Boolean function of n variables is a special case of a symmetric function of $2^n - 1$ variables. [See W. H. Kautz, *IRE Transactions* **EC-10** (1961), 378.]

115. Both sides are self-dual, so we may assume that $x_0 = 0$. Then

$$s_j = [x_j + \cdots + x_{j+m-1} > x_{j+m} + \cdots + x_{j+2m-1}].$$

If $x_1 + \cdots + x_{2m}$ is odd, we have $s_j = \bar{s}_{j+m}$; hence $s_1 + \cdots + s_{2m} = m$ and the result is 1. But if $x_1 + \cdots + x_{2m}$ is even, the difference $x_j + \cdots + x_{j+m-1} - x_{j+m} - \cdots - x_{j+2m-1}$ will be zero for at least one $j \le m$; that makes $s_j = s_{j+m} = 0$, so we will have $s_1 + \cdots + s_{2m} < m$.

116. (a) It's an implicant if and only if $f(x) = 1$ whenever $j \le \nu x \le n - k + j$. It's a prime implicant if and only if we also have $f(x) = 0$ when $\nu x = j-1$ or $\nu x = n-k+j+1$.

(b) Consider the string $v = v_0 v_1 \ldots v_n$ such that $f(x) = v_{\nu x}$. By part (a), there are $\binom{a+b+c}{a,b,c}$ prime implicants when $v = 0^a 1^{b+1} 0^c$. In the stated case, $a = b = c = 3$, so there are 1680 prime implicants.

(c) For a general symmetric function, we add together the prime implicants for each run of 1s in v. Clearly there are more for $v = 0^{a+1}1^{b+1}0^{c-1}$ than for $v = 0^a 1^{b+1} 0^c$ when $a < c-1$; so v contains no two consecutive 0s when the maximum is reached.

Let $\hat{b}(m,n)$ be the maximum number of prime implicants possible when $v_m = 1$ and $v_j = 0$ for $m < j \le n$. Then when $m \le \frac{1}{2}n$ we have

$$\hat{b}(m,n) = \max_{0\le k\le m}\left(\binom{n}{k, m-k, n-m} + \hat{b}(k-2,n)\right) = \binom{n}{\lceil m/2\rceil, \lfloor m/2\rfloor, n-m} + \hat{b}(\lceil m/2\rceil - 2, n),$$

with $\hat{b}(-2,n) = \hat{b}(-1,n) = 0$. And the overall maximum is

$$\hat{b}(n) = \binom{n}{n_0, n_1, n_2} + \hat{b}(n_1 - 2, n) + \hat{b}(n_2 - 2, n), \qquad n_j = \left\lfloor \frac{n+j}{3} \right\rfloor.$$

In particular we have $\hat{b}(9) = 1698$, with the maximum occurring for $v = 1101111011$.

(d) By Stirling's approximation, $\hat{b}(n) = 3^{n+3/2}/(2\pi n) + O(3^n/n^2)$.

(e) In this case the appropriate recurrence for $m < \lceil n/2 \rceil$ is

$$\tilde{b}(m,n) = \max_{0\le k\le m}\left(\binom{n}{k, m-k, n-m} + \binom{n}{k-1, 0, n-k+1} + \tilde{b}(k-2,n)\right) = \binom{n}{\lceil m/2\rceil, \lfloor m/2\rfloor, n-m} + \binom{n}{\lceil m/2\rceil - 1} + \tilde{b}(\lceil m/2\rceil - 2, n)$$

and $\tilde{b}(n) = \tilde{b}(\lceil n/2\rceil - 1, n)$ maximizes min(prime implicants(f), prime implicants($\bar{f}$)). We have $(\tilde{b}(1), \tilde{b}(2), \ldots) = (1, 1, 4, 5, 21, 31, 113, 177, 766, 1271, 4687, 7999, 34412, \ldots)$; for example, $\tilde{b}(9) = 766$ corresponds to $S_{0,2,3,4,8}(x_1, \ldots, x_9)$. Asymptotically, $\tilde{b}(n) = 2^{(3n+3+(n \bmod 2))/2}/(2\pi n) + O(2^{3n/2}/n^2)$.

References: *Summaries, Summer Inst. for Symbolic Logic* (Dept. of Math., Cornell Univ., 1957), 211–212; B. Dunham and R. Fridshal, *J. Symbolic Logic* **24** (1959), 17–19; A. P. Vikulin, *Problemy Kibernetiki* **29** (1974), 151–166, which reports on work done in 1960; Y. Igarashi, *Transactions of the IEICE of Japan* **E62** (1979), 389–394.

117. The maximum number of subcubes of the n-cube, with none contained in another, is obtained when we choose all subcubes of dimension $\lfloor n/3 \rfloor$. (It is also obtained by choosing all subcubes of dimension $\lfloor (n+1)/3 \rfloor$; for example, when $n = 2$ we can choose either $\{0{*}, 1{*}, {*}0, {*}1\}$ or $\{00, 01, 10, 11\}$.) Hence $b^*(n) = \binom{n}{\lfloor n/3\rfloor} 2^{n-\lfloor n/3\rfloor} = 3^{n+1}/\sqrt{4\pi n} + O(3^n/n^{3/2})$. [See the paper of Vikulin in the previous answer, pages 164–166; A. K. Chandra and G. Markowsky, *Discrete Math.* **24** (1978), 7–11; N. Metropolis and G. C. Rota, *SIAM J. Applied Math.* **35** (1978), 689–694.]

118. Consider two functions equivalent if we can obtain one from the other by complementing and/or permuting variables, but not complementing the function value itself. Such functions clearly have the same number of prime implicants; this equivalence relation is studied further in answer 125 below. A computer program based on exercise 30 produces the following results:

m	Classes	Functions	m	Classes	Functions	m	Classes	Functions
0	1	1	5	87	17472	10	7	632
1	5	81	6	70	12696	11	1	96
2	18	1324	7	43	7408	12	2	24
3	46	6608	8	24	3346	13	1	16
4	87	14536	9	10	1296	14	0	0

And here are the corresponding statistics for functions of five variables:

m	Classes	Functions	m	Classes	Functions	m	Classes	Functions
0	1	1	11	186447	666555696	22	338	608240
1	6	243	12	165460	590192224	23	130	197440
2	37	14516	13	129381	459299440	24	71	75720
3	244	318520	14	91026	319496560	25	37	28800
4	1527	3319580	15	57612	199792832	26	15	10560
5	6997	19627904	16	33590	113183894	27	6	2880
6	23434	73795768	17	17948	58653984	28	4	1040
7	57048	190814016	18	8880	27429320	29	2	640
8	105207	362973410	19	3986	11597760	30	2	48
9	152763	538238660	20	1795	4548568	31	2	64
10	183441	652555480	21	720	1633472	32	1	16

119. Several authors have conjectured that $b(n) = \hat{b}(n)$; M. M. Gadzhiev has proved that equality holds for $n \le 6$ [*Diskretnyĭ Analiz* **18** (1971), 3–24].

120. (a) Every prime implicant is a minterm, since no adjacent points of the n-cube have the same parity. So the full disjunctive form is the only decent DNF in this case.

(b) Now all prime implicants consist of two adjacent points. We must include the 14 subcubes $0^j{*}0^{6-j}$ and $1^j{*}1^{6-j}$ for $0 \le j \le 6$, in order to cover the points with $\nu x = 1$ and $\nu x = 6$. The other $\binom{7}{3} + \binom{7}{4} = 70$ points can be covered by 35 well-chosen prime implicants (see, for example, exercise 6.5–1, or the "Christmas tree pattern" in Section 7.2.1.6). Thus the shortest DNF has length 49. [An ingeniously plausible but fallacious argument that 70 prime implicants are necessary was presented by S. B. Yablonsky in *Problemy Kibernetiki* **7** (1962), 229–230.]

(c) For each of 2^{n-1} choices of $(x_1, \ldots, x_{n-1})$ we need at most one implicant to account for the behavior of the function with respect to x_n.

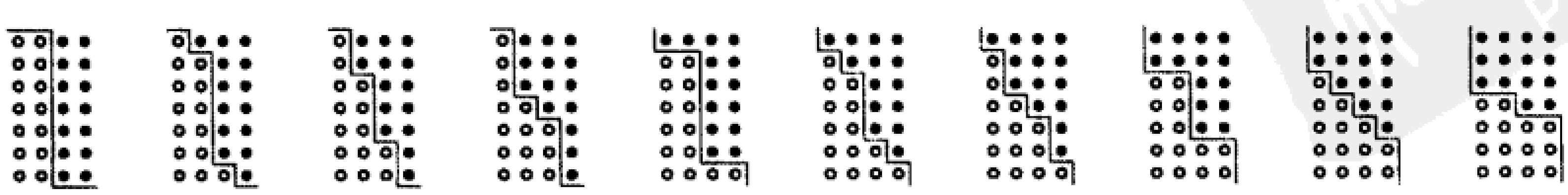
[Asymptotically, almost all Boolean functions of n variables have a shortest DNF with $\Theta(2^n/(\log n \log\log n))$ prime implicants. See R. G. Nigmatullin, *Diskretnyĭ Analiz* **10** (1967), 69–89; V. V. Glagolev, *Problemy Kibernetiki* **19** (1967), 75–94; A. D. Korshunov, *Metody Diskretnogo Analiza* **37** (1981), 9–41; N. Pippenger, *Random Structures & Algorithms* **22** (2003), 161–186.]

121. (a) Let $x = x_1 \ldots x_m$ and $y = y_1 \ldots y_n$. Since f is a function of $(\nu x, \nu y)$, there are altogether $2^{(m+1)(n+1)}$ possibilities.

(b) In this case $\nu x \le \nu x'$ and $\nu y \le \nu y'$ implies $f(x,y) \le f(x',y')$. Every such function corresponds to a zigzag path from $a_0 = (-\frac{1}{2}, n+\frac{1}{2})$ to $a_{m+n+2} = (m+\frac{1}{2}, -\frac{1}{2})$, with $a_j = a_{j-1} + (1,0)$ or $a_j = a_{j-1} - (0,1)$ for $1 \le j \le m+n+2$; we have $f(x,y) = 1$ if and only if the point $(\nu x, \nu y)$ lies above the path. So the number of possibilities is the number of such paths, namely $\binom{m+n+2}{m+1}$.

(c) Complementing x and y changes νx to $m - \nu x$ and νy to $n - \nu y$. So there are no such functions when m and n are both even; otherwise there are $2^{(m+1)(n+1)/2}$.

(d) The path in (b) must now satisfy $a_j + a_{m+n+2-j} = (m,n)$ for $0 \le j \le m+n+2$. Hence there are $\binom{\lceil m/2\rceil + \lceil n/2\rceil}{\lceil m/2\rceil}[m \text{ odd or } n \text{ odd}]$ such functions. For example, the ten cases when $m = 3$ and $n = 6$ are



122. A function of this kind is regular with the x's to the left of the y's if and only if the zigzag path does not contain two points (x,y) and $(x+2,y)$ with $0<y<n$; it is regular with the y's left of the x's if and only if the zigzag path does not contain both $(x,y+2)$ and (x,y) with $0<x<m$. It is a threshold function if and only if there is a straight line through the point $(m/2,n/2)$ with the property that (s,t) is above the line if and only if (s,t) is above the path, for $0\le s\le m$ and $0\le t\le n$. So cases 5 and 8, illustrated in the previous answer, fail to be regular; cases 1, 2, 3, 7, 9, and 10 are threshold functions. The regular non-threshold functions that remain can also be expressed as follows: $((x_1\vee x_2\vee x_3)\wedge\langle x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle)\vee(x_1\wedge x_2\wedge x_3)$ (case 4); $\langle 00x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle\vee(\langle x_1x_2x_3\rangle\wedge\langle 11x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle)$ (case 6).

123. Self-dual *regular* functions are relatively easy to list, for small n, but the numbers grow rapidly: When $n=9$ there are 319,124 of them, found by Muroga, Tsuboi, and Baugh in 1967, and when $n=10$ there are 1,214,554,343 (see exercise 7.1.4–00). The corresponding numbers for $n\le 6$ appear in Table 5, because all such functions are threshold functions when $n<9$; there are 135 when $n=7$, and 2470 when $n=8$.

The threshold condition can be tested quickly for any such function by improving on the method of exercise 103, because constraints are needed only for the *minimal* vectors x (with respect to majorization) such that $f(x)=1$.

The number θ_n of n-variable threshold functions is known to satisfy $\lg\theta_n = n^2-O(n^2/\log n)$; see Yu. A. Zuev, *Matematicheskie Voprosy Kibernetiki* **5** (1994), 5–61.

124. The 222 equivalence classes listed in Table 5 include 24 classes of size $2^{n+1}n! = 768$; so there are $24\times 768 = 18432$ answers to this problem. One of them is the function $(w\wedge(x\vee(y\wedge z)))\oplus z$.

125. 0; x; $x\wedge y$; $x\wedge y\wedge z$; $x\wedge(y\vee z)$; $x\wedge(y\oplus z)$. (These functions are $x\wedge f(y,z)$, where f runs through the equivalence classes of two-variable functions under permutation and/or complementation of variables but *not* of the function values. In general, let $f\simeq g$ mean that f is equivalent to g in that weaker sense, but write $f\cong g$ if they are equivalent in the sense of Table 5. Then $x\wedge f\cong x\wedge g$ if and only if $f\simeq g$, assuming that f and g are independent of the variable x. For it's easy to see that $(x\wedge f)\simeq(\bar x\vee\bar g)$ is impossible. And if $(x\wedge f)\simeq(x\wedge g)$, we can prove that $f\simeq g$ by showing that, if σ is a signed permutation of $\{x_0,\ldots,x_n\}$ and if $x=x_1\ldots x_n$, then the identity $x_0\wedge f(x) = (x_0\sigma)\wedge g(x\sigma)$ implies $f(x)=g(x\sigma\tau)$, where τ interchanges $x_0\leftrightarrow x_0\sigma$. Consequently the bottom line of Table 5 enumerates equivalence classes under $\simeq$, but with n increased by 1; there are, for example, 402 such classes of 4-variable functions.)

126. (a) The function is canalizing if and only if it has a prime implicant with at most one literal, or a prime clause with at most one literal.

(b) The function is canalizing if and only if at least one of the components of $\Sigma(f)$ is equal to 0, 2^{n-1}, $N(f)$, or $N(f)-2^{n-1}$. [See I. Shmulevich, H. Lähdesmäki, and K. Egiazarian, *IEEE Signal Processing Letters* **11** (2004), 289–292, Proposition 6.]

(c) If, say, $\vee(f) = y_1\ldots y_n$ with $y_j=0$, then $f(x)=0$ whenever $x_j=1$. Therefore f is canalizing if and only if we don't have $\vee(f)=\vee(\bar f)=1\ldots 1$ and $\wedge(f)=\wedge(\bar f)=0\ldots 0$. With this test one can prove that many functions are noncanalizing when their value is known at only a few points.

127. (a) Since a self-dual function $f(x_1,\ldots,x_n)$ is true at exactly 2^{n-1} points, it is canalizing with respect to the variable x_j if and only if $f(x_1,\ldots,x_n)=x_j$.

(b) A definite Horn function is clearly canalizing if (i) it contains any clause with a single literal, or (ii) some literal occurs in every clause. Otherwise it is not canalizing. For we have $f(0,\ldots,0)=f(1,\ldots,1)=1$, because (i) is false; and if x_j is any variable,

there is a clause C_0 not containing $\bar{x}_j$ and a clause C_1 not containing x_j, because (ii) is false. By choosing appropriate values of the other variables, we can make $C_0 \wedge C_1$ false when $x_j = 0$ and also when $x_j = 1$.

128. For example, $(x_1 \wedge \cdots \wedge x_n) \vee (\bar{x}_1 \wedge \cdots \wedge \bar{x}_n)$.

129. $\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}2^{2^{n-k}+k+1} - 2(n-1) - 4(n \bmod 2) = n2^{2^{n-1}+2} + O(n^2 2^{2^{n-2}})$. [See W. Just, I. Shmulevich, and J. Konvalina, *Physica* **D197** (2004), 211–221.]

130. (a) If there are a_n functions of n or fewer variables, but b_n functions of exactly n variables, we have $a_n = \sum_k \binom{n}{k} b_k$. Therefore $b_n = \sum_k (-1)^{n-k}\binom{n}{k}a_k$. (This rule, noted by C. E. Shannon in *Trans. Amer. Inst. Electrical Engineers* **57** (1938), 713–723, §4, applies to all rows of Table 3, *except* for the case of symmetric functions.) In particular, the answer sought here is $168 - 4\cdot 20 + 6\cdot 6 - 4\cdot 3 + 2 = 114$.

(b) If there are a'_n essentially distinct functions of n or fewer variables, and b'_n of exactly n variables, we have $a'_n = \sum_{k=0}^{n} b'_k$. Hence $b'_n = a'_n - a'_{n-1}$, and the answer in this case is $30 - 10 = 20$.

131. Let there be $h(n)$ Horn functions and $k(n)$ Krom functions. Clearly $\lg h(n) \ge \binom{n}{\lfloor n/2 \rfloor}$ and $\lg k(n) \ge \binom{n}{2}$. V. B. Alekseyev [*Diskretnaîa Matematika* **1** (1989), 129–136] has proved that $\lg h(n) = \binom{n}{\lfloor n/2\rfloor}(1 + O(n^{-1/4}\log n))$. B. Bollobás, G. Brightwell, and I. Leader [*Israel J. Math.* **133** (2003), 45–60] have proved that $\lg k(n) \sim \frac{1}{2}n^2$.

132. (a) The hint is true because $\sum_y s(y)s(y\oplus z) = \sum_{w,x,y}(-1)^{f(w)+w\cdot y+f(x)+x\cdot(y+z)} = 2^n\sum_{w,x}(-1)^{f(w)+f(x)+x\cdot z}[x=w]$. Now suppose that $f(x) = g(x)$ for $2^{n-1}+k$ values of x; then $f(x) = g(x)\oplus 1$ for $2^{n-1}-k$ values of x. But if $|k| < 2^{n/2-1}$ for all affine g, we would have $|s(y)| < 2^{n/2}$ for all y, contradicting the hint when $z = 0$.

(b) Given $y = y_1\ldots y_n$, there are exactly $2^{n/2}((y_1y_2 + y_3y_4 + \cdots + y_{n-1}y_n + 1 + h(y_1, y_3, \ldots, y_{n-1})) \bmod 2)$ solutions to $f(x) = x\cdot y \bmod 2$ when $x_{2k} = y_{2k-1}$ for $1 \le k \le n/2$, and there are $2^{n/2-1}$ solutions for each of the other $2^{n/2}-1$ values of $(x_2, x_4, \ldots, x_n)$. So there are $2^{n-1} \pm 2^{n/2-1}$ solutions altogether. (This argument proves, in fact, that $(g(x_1, x_3, \ldots, x_{2n-1})\cdot(x_2, x_4, \ldots, x_{2n}) + h(x_2, x_4, \ldots, x_{2n})) \bmod 2$ is bent whenever $g(x_1, x_3, \ldots, x_{2n-1})$ is a permutation of all $2^{n/2}$-bit vectors.)

(c) If $f(x)$ is bent, the argument in part (a) proves that $s(y) = 2^{n/2}(-1)^{g(y)}$ for some Boolean function $g(y)$. This function is also bent, because $\sum_y (-1)^{g(y)+w\cdot y} = 2^{-n/2}\sum_{x,y}(-1)^{f(x)+x\cdot y+w\cdot y} = 2^{n/2}\sum_x(-1)^{f(x)}[x=w] = 2^{n/2}(-1)^{f(w)}$ for all w. Consequently the hint tells us that we have $\sum_x(-1)^{f(x)+f(x\oplus z)} = 0$ for all nonzero z.

Conversely, assume that $f(x)$ satisfies the stated condition, and that $f(x) = 1$ for exactly k values of x. If y is any Boolean vector, the function $g(x) = (f(x)+x\cdot y) \bmod 2$ also satisfies the condition; therefore we need only prove that $k = 2^{n-1} \pm 2^{n/2-1}$.

Let A be the $2^n\times 2^n$ matrix with $f(x\oplus y)$ in row x and column y, and let J be the $2^n \times 2^n$ matrix of all 1s. Then we have $J^2 = 2^nJ$ and $JA = AJ = kJ$, hence

$$2^nI \;=\; (J-2A)^2 \;=\; 2^nJ - 4kJ + 4A^2;$$

and it follows, when $y \ne 0\ldots 0$, that the number of x such that $f(x) = 1$ and $f(x\oplus y) = 1$ is $k - 2^{n-2}$. In other words, there are exactly $k - 2^{n-2}$ ordered pairs of vectors (w, x) such that $f(w) = f(x) = 1$ and $w\oplus x = y$. Summing over all y gives $(2^n-1)(k-2^{n-2}) = k(k-1)$, and the solutions to this quadratic equation are $k = 2^{n-1} \pm 2^{n/2-1}$.

(d) By exercise 11, the term $x_1\ldots x_r$ is present if and only if the equation $f(x_1, \ldots, x_r, 0, \ldots, 0) = 1$ has an odd number of solutions, and an equivalent condition

is $\left(\sum_{x_1,\ldots,x_r}(-1)^{f(x_1,\ldots,x_r,0,\ldots,0)}\right) \bmod 4 = 2$. We've seen in part (c) that this sum is

$$2^{-n}\sum_{x_1,\ldots,x_r,y} s(y)(-1)^{x_1y_1+\cdots+x_ry_r} = 2^{r-n}\sum_{y_{r+1},\ldots,y_n} s(0,\ldots,0,y_{r+1},\ldots,y_n).$$

If $r = n$, the latter sum is $\pm 2^{n/2}$; otherwise it contains an even number of summands, each of which is $\pm 2^{r-n/2}$. So the result is a multiple of 4.

[Bent functions were introduced by O. S. Rothaus in 1966; his privately circulated paper was eventually published in *J. Combinatorial Theory* **A20** (1976), 300–305. J. F. Dillon, *Congressus Numerantium* **14** (1975), 237–249, discovered additional families of bent functions, and many other examples have subsequently been found when $n \geq 8$ and n is even. Bent functions don't exist when n is odd, but a function like $g(x_1,\ldots,x_{n-1}) \oplus x_n \wedge h(x_1,\ldots,x_{n-1})$ has distance $2^{n-1} - 2^{(n-3)/2}$ from all affine functions when g and h are bent. A better construction for the case $n = 15$ was found by N. J. Patterson and D. H. Wiedemann, *IEEE Transactions* **IT-29** (1983), 354–356, **IT-36** (1990), 443, achieving distance $2^{14} - 108$.]

133. Let $p_k = 1/(2^{2^{n-k}}+1)$, so that $\bar{p}_k = 2^{2^{n-k}}/(2^{2^{n-k}}+1)$. [Ph.D. thesis (MIT, 1994).]

SECTION 7.1.2

1. $((x_1 \vee x_4) \wedge x_2) \equiv (x_1 \vee x_3)$.

2. (a) $(w \oplus (x \wedge y)) \oplus ((x \oplus y) \wedge z)$; (b) $(w \wedge (x \vee y)) \wedge ((x \wedge y) \vee z)$.

3. [*Doklady Akademii Nauk SSSR* **115** (1957), 247–248.] Construct a $k \times n$ matrix whose rows are the vectors x where $f(x) = 1$. By permuting and/or complementing variables, we may assume that the top row is $1\ldots1$ and that the columns are sorted. Suppose there are l distinct columns. Then $f = g \wedge h$, where g is the AND of the expressions $(x_{j-1} \equiv x_j)$ over all $1 < j \le n$ such that column $j-1$ equals column j, and h is the OR of k minterms of length l, using one variable from each group of equal columns. For example, if $n = 8$ and if f is 1 at the $k = 3$ points 11111111, 00001111, 00110111, then $l = 4$ and $f(x)$ equals $(x_1 \equiv x_2) \wedge (x_3 \equiv x_4) \wedge (x_6 \equiv x_7) \wedge (x_7 \equiv x_8) \wedge ((x_1 \wedge x_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge x_3 \wedge \bar{x}_5 \wedge x_6))$. The length of this formula in general is $2n + (k-2)l - 1$, and we have $l \le 2^{k-1}$.

Notice that, if k is large, we get shorter formulas by writing $f(x)$ as a disjunction $f_1(x) \vee \cdots \vee f_r(x)$, where each f_j has at most $\lceil k/r \rceil$ 1s. Thus

$$L(f) \le \min_{r \ge 1}\left(r - 1 + (2n + \lceil k/r - 2 \rceil 2^{\lceil k/r - 1 \rceil})r\right).$$

4. The first inequality is obvious, because a binary tree of depth d has at most $1 + 2 + \cdots + 2^{d-1} = 2^d - 1$ internal nodes.

The hint follows because we can find a minimal subtree of size $\ge \lfloor r/3 \rfloor$. Its size s is at most $1 + 2(\lfloor r/3 \rfloor - 1)$. Therefore we can write $f = (g?\ f_1\!:\ f_0)$, where g is a subformula of size s; f_0 and f_1 are the formulas of size $r - s - 1$ obtained when that subformula is replaced by 0 and 1, respectively.

Let $d(r) = \max\{\, D(f) \mid L(f) = r \,\}$. Since the mux function has depth 2, and since $\max(s, r - s - 1) < \lceil \frac{2r}{3} \rceil$, we have $d(r) \le 2 + d(\lceil \frac{2r}{3} \rceil - 1)$ for $r \ge 3$, and the result follows by induction on r. [*Hawaii International Conf. System Sci.* **4** (1971), 525–527.]

5. Let $g_0 = 0$, $g_1 = x_1$, and $g_j = x_j \wedge (x_{j-1} \vee g_{j-2})$ for $j \ge 2$. Then $F_n = g_n \vee g_{n-1}$, with cost $2n - 2$ and depth n. [These functions g_j also play a prominent role in binary addition; see exercises 42 and 44 for ways to compute them with depth $O(\log n)$.]

6. True: Consider the cases $y = 0$ and $y = 1$.

7. $\hat{x}_5 = x_1 \vee x_4$, $\hat{x}_6 = x_2 \wedge \hat{x}_5$, $\hat{x}_7 = x_1 \vee x_3$, $\hat{x}_8 = \hat{x}_6 \oplus \hat{x}_7$. (The original chain computes the "random" function (6); see exercise 1. The new chain computes the normalization of that function, namely its complement.)

8. The desired truth table consists of blocks of 2^{n-k} 0s alternating with blocks of 2^{n-k} 1s, as in (7). Therefore, if we multiply by $2^{2^{n-k}} + 1$ we get $x_k + (x_k \ll 2^{n-k})$, which is all 1s.

9. When finding $L(f) = \infty$ in step L6, we can store g and h in a record associated with f. Then a recursive procedure will be able to construct a minimum-length formula for f from the respective formulas for g and h.

10. In step L3, use $k = r - 1$ instead of $k = r - 1 - j$. Also change L to D everywhere.

11. The only subtle point is that j should *decrease* in step U3; then we'll never have $\phi(g) \,\&\, \phi(h) \ne 0$ when $j = 0$, so all cases of cost $r - 1$ will be discovered before we begin to look at list $r - 1$.

U1. [Initialize.] Set $U(0) \leftarrow \phi(0) \leftarrow 0$ and $U(f) \leftarrow \infty$ for $1 \le f < 2^{2^n - 1}$. Then set $U(x_k) \leftarrow \phi(x_k) \leftarrow 0$ and put x_k into list 0, as in step L1. Also set

$U(x_j \circ x_k) \leftarrow 1$, set $\phi(x_j \circ x_k)$ to an appropriate bit vector of weight 1, and put $x_j \circ x_k$ into list 1, for $1 \le j < k \le n$ and all five normal operators $\circ$. Finally set $c \leftarrow 2^{2^n-1} - 5\binom{n}{2} - n - 1$.

U2. [Loop on r.] Do step U3 for $r = 2, 3, \ldots,$ while $c > 0$.

U3. [Loop on j and k.] Do step U4 for $j = \lfloor (r-1)/2 \rfloor, \lfloor (r-1)/2 \rfloor - 1, \ldots,$ and $k = r - 1 - j$, while $j \ge 0$.

U4. [Loop on g and h.] Do step U5 for all g in list j and all h in list k; if $j = k$, restrict h to functions that *follow* g in list k.

U5. [Loop on f.] If $\phi(g) \,\&\, \phi(h) \ne 0$, set $u \leftarrow r - 1$ and $v \leftarrow \phi(g) \,\&\, \phi(h)$; otherwise set $u \leftarrow r$ and $v \leftarrow \phi(g) \mid \phi(h)$. Then do step U6 for $f = g \,\&\, h$, $f = \bar{g} \,\&\, h$, $f = g \,\&\, \bar{h}$, $f = g \mid h$, and $f = g \oplus h$.

U6. [Update $U(f)$ and $\phi(f)$.] If $U(f) = \infty$, set $c \leftarrow c - 1$, $\phi(f) \leftarrow v$, and put f into list u. Otherwise if $U(f) > u$, set $\phi(f) \leftarrow v$ and move f from list $U(f)$ to list u. Otherwise if $U(f) = u$, set $\phi(f) \leftarrow \phi(f) \mid v$. ▌

12. $x_4 = x_1 \oplus x_2$, $x_5 = x_3 \wedge x_2$, $x_6 = x_2 \wedge \bar{x}_4$, $x_7 = x_5 \vee x_6$.

13. $f_5 = 01010101$ (x_3); $f_4 = 01110111$ $(x_2 \vee x_3)$; $f_3 = 01110101$ $((\bar{x}_1 \wedge x_2) \vee x_3)$; $f_2 = 00110101$ $(x_1?\ x_3{:}\ x_2)$; $f_1 = 00010111$ $(\langle x_1x_2x_3 \rangle)$.

14. For $1 \le j \le n$, first compute $t \leftarrow (g \oplus (g \gg 2^{n-j})) \,\&\, x_j$, $t \leftarrow t \oplus (t \ll 2^{n-j})$, where x_j is the truth table (11); then for $1 \le k \le n$ and $k \ne j$, the desired truth table corresponding to $x_j \leftarrow x_j \circ x_k$ is $g \oplus (t \,\&\, ((x_j \circ x_k) \oplus x_j))$.

(The $5n(n-1)$ masks $(x_j \circ x_k) \oplus x_j$ are independent of g and can be computed in advance. The same idea applies if we allow more general computations of the form $x_{j(i)} \leftarrow x_{k(i)} \circ_i x_{l(i)}$, with $5n^2(n-1)$ masks $(x_k \circ x_l) \oplus x_j$.)

15. Remarkably asymmetrical ways to compute symmetrical functions:

(a)	(b)	(c)
$x_1 \leftarrow x_1 \oplus x_2$,	$x_1 \leftarrow x_1 \oplus x_2$,	$x_1 \leftarrow x_1 \oplus x_2$,
$x_3 \leftarrow x_3 \oplus x_4$,	$x_2 \leftarrow x_2 \wedge \bar{x}_1$,	$x_2 \leftarrow x_2 \oplus x_3$,
$x_1 \leftarrow x_1 \oplus x_3$,	$x_3 \leftarrow x_3 \oplus x_4$,	$x_2 \leftarrow x_2 \vee x_1$,
$x_2 \leftarrow x_2 \oplus x_4$,	$x_4 \leftarrow x_4 \wedge x_1$,	$x_1 \leftarrow x_1 \oplus x_4$,
$x_3 \leftarrow x_3 \vee x_2$,	$x_2 \leftarrow \bar{x}_2 \wedge x_3$,	$x_1 \leftarrow x_1 \wedge x_3$,
$x_3 \leftarrow x_3 \wedge \bar{x}_1$.	$x_2 \leftarrow x_2 \oplus x_1$,	$x_2 \leftarrow x_2 \wedge \bar{x}_1$,
	$x_2 \leftarrow x_2 \wedge \bar{x}_4$.	$x_2 \leftarrow x_2 \oplus x_4$.

16. A computation that uses only $\oplus$ and complementation produces nothing but affine functions (see exercise 7.1.1–132). Suppose $f(x) = f(x_1, \ldots, x_n)$ is a non-affine function computable in minimum memory. Then $f(x)$ has the form $g(Ax + c)$ where $g(y_1, y_2, \ldots, y_n) = g(y_1 \wedge y_2, y_2, \ldots, y_n)$, for some nonsingular $n \times n$ matrix A of 0s and 1s, where x and c are column vectors and the vector operations are performed modulo 2; in this formula the matrix A and vector c account for all operations $x_i \leftarrow x_i \oplus x_j$ and/or permutations and complementations of coordinates that occur after the most recent non-affine operation that was performed. We will exploit the fact that $g(0, 0, y_3, \ldots, y_n) = g(1, 0, y_3, \ldots, y_n)$.

Let α and β be the first two rows of A; also let a and b be the first two elements of c. Then if $Ax + c \equiv y$ (modulo 2) we have $y_1 = y_2 = 0$ if and only if $\alpha \cdot x \equiv a$ and $\beta \cdot x \equiv b$. Exactly 2^{n-2} vectors x satisfy this condition, and for all such vectors we have $f(x) = f(x \oplus w)$, where $Aw \equiv (1, 0, \ldots, 0)^T$.

Given α, β, a, b, and w, with $\alpha \ne (0, \ldots, 0)$, $\beta \ne (0, \ldots, 0)$, $\alpha \ne \beta$, and $\alpha \cdot w \equiv 1$ (modulo 2), there are $2^{2^n - 2^{n-2}}$ functions f with the property that $f(x) = f(x \oplus w)$

whenever $\alpha \cdot x \bmod 2 = a$ and $\beta \cdot x \bmod 2 = b$. Therefore the total number of functions computable in minimum memory is at most 2^{n+1} (for affine functions) plus

$$(2^n - 1)(2^n - 2)2^2(2^{n-1})(2^{2^n - 2^{n-2}}) < 2^{2^n - 2^{n-2} + 3n + 1}.$$

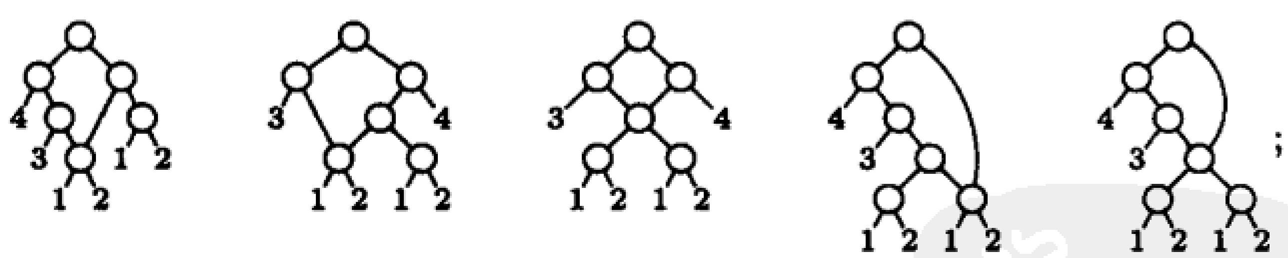
17. Let $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{n-1}) \oplus (h(x_1, \ldots, x_{n-1}) \wedge x_n)$ as in 7.1.1–(16). Representing h in CNF, form the clauses one by one in x_0 and AND them into x_n, obtaining $h \wedge x_n$. Representing g as a sum (mod 2) of conjunctions, form the successive conjunctions in x_0 and XOR them into x_n when ready.

(It appears to be impossible to evaluate all functions inside of $n+1$ registers if we disallow the non-canalizing operators $\oplus$ and $\equiv$. But $n+2$ registers clearly do suffice, even if we restrict ourselves to the single operator $\overline{\wedge}$.)

18. As mentioned in answer 14, we should extend the text's definition of minimum-memory computation to allow also steps like $x_{j(i)} \leftarrow x_{k(i)} \circ_i x_{l(i)}$, with $k(i) \neq j(i)$ and $l(i) \neq j(i)$, because that will give better results for certain functions that depend on only four of the five variables. Then we find $C_m(f) = (0, 1, \ldots, 13, 14)$ for respectively (2, 2, 5, 20, 93, 389, 1960, 10459, 47604, 135990, 198092, 123590, 21540, 472, 0) classes of functions ... leaving 75,908 classes (and 575,963,136 functions) for which $C_m(f) = \infty$ because they cannot be evaluated *at all* in minimum memory. The most interesting function of that kind is probably $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4) \vee (x_4 \wedge x_5) \vee (x_5 \wedge x_1)$, which has $C(f) = 7$ but $C_m(f) = \infty$. Another interesting case is $(((x_1 \vee x_2) \oplus x_3) \vee ((x_2 \vee \bar{x}_4) \wedge x_5)) \wedge ((x_1 \equiv x_2) \vee x_3 \vee x_4)$, for which $C(f) = 8$ and $C_m(f) = 13$. One way to evaluate that function in eight steps is $x_6 = x_1 \vee x_2$, $x_7 = x_1 \vee x_4$, $x_8 = x_2 \oplus x_7$, $x_9 = x_3 \oplus x_6$, $x_{10} = x_4 \oplus x_9$, $x_{11} = x_5 \vee x_9$, $x_{12} = x_8 \wedge x_{10}$, $x_{13} = x_{11} \wedge \bar{x}_{12}$.

19. If not, the left and right subtrees of the root must overlap, since case (i) fails. Each variable must occur at least once as a leaf, by hypothesis. At least two variables must occur at least twice as leaves, since case (ii) fails. But we can't have $n+2$ leaves with $r \leq n+1$ internal nodes, unless the subtrees fail to overlap.

20. Now Algorithm L (with '$f = g \oplus h$' omitted in step L5) shows that some formulas must have length 15; and even the footprint method of exercise 11 does no better than 14. To get truly minimum chains, the 25 special chains for $r = 6$ in the text must be supplemented by five others that can no longer be ruled out, namely



and when $r = (7, 8, 9)$ we must also consider respectively (653, 12387, 225660) additional potential chains that are not special cases of the top-down and bottom-up constructions. Here are the resulting statistics, for comparison with Table 1:

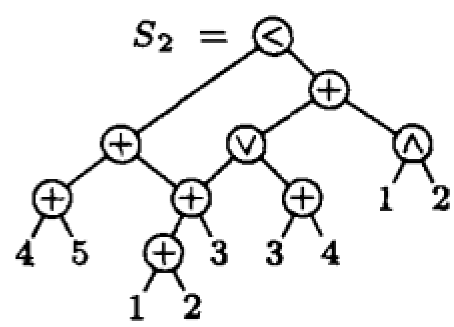
$C_c(f)$	Classes	Functions	$U_c(f)$	Classes	Functions	$L_c(f)$	Classes	Functions	$D_c(f)$	Classes	Functions
0	2	10	0	2	10	0	2	10	0	2	10
1	1	48	1	1	48	1	1	48	1	1	48
2	2	256	2	2	256	2	2	256	2	7	684
3	7	940	3	7	940	3	7	940	3	59	17064
4	9	2336	4	9	2336	4	7	2048	4	151	47634
5	24	6464	5	21	6112	5	20	5248	5	2	96
6	30	10616	6	28	9664	6	23	8672	6	0	0

7	61	18984	7	45	15128	7	37	11768	7	0	0
8	45	17680	8	40	14296	8	27	10592	8	0	0
9	37	7882	9	23	8568	9	33	11536	9	0	0
10	4	320	10	28	5920	10	16	5472	10	0	0
11	0	0	11	6	1504	11	30	6304	11	0	0
12	0	0	12	5	576	12	3	960	12	0	0
13	0	0	13	3	144	13	8	1472	13	0	0
14	0	0	14	2	34	14	2	96	14	0	0
15	0	0	15	0	0	15	4	114	15	0	0

The two function classes of depth 5 are represented by $S_{2,4}(x_1,x_2,x_3,x_4)$ and $x_1 \oplus S_2(x_2,x_3,x_4)$; and those two functions, together with $S_2(x_1,x_2,x_3,x_4)$ and the parity function $S_{1,3}(x_1,x_2,x_3,x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, have length 15. Also $U_c(S_{2,4}) = U_c(S_{1,3}) = 14$. The four classes of cost 10 are represented by $S_{1,4}(x_1,x_2,x_3,x_4)$, $S_{2,4}(x_1,x_2,x_3,x_4)$, $(x_4?\ x_1 \oplus x_2 \oplus x_3\colon \langle x_1x_2x_3\rangle)$, and $[(x_1x_2x_3x_4)_2 \in \{0,1,4,7,10,13\}]$. (The third of these, incidentally, is the complement of (20), "Harvard's hardest case.")

21. (The authors stated that their table entries "should be regarded only as the most economical operators known to the present writers.")

22. $\nu(x_1x_2x_3x_4x_5) = 3$ if and only if $\nu(x_1x_2x_3x_4) \in \{2,3\}$ and $\nu(x_1x_2x_3x_4x_5)$ is odd. Similarly, $S_2(x_1,x_2,x_3,x_4,x_5) = S_3(\bar{x}_1,\bar{x}_2,\bar{x}_3,\bar{x}_4,\bar{x}_5)$ incorporates $S_{1,2}(x_1,x_2,x_3,x_4)$:



23. We need only consider the 32 normal cases, as in Fig. 9, since the complement of a symmetric function is symmetric. Then we can use reflection, like $S_{1,2}(x) = S_{3,4}(\bar{x})$, possibly together with complementation, like $S_{2,3,4,5}(x) = \bar{S}_{0,1}(x) = \bar{S}_{4,5}(\bar{x})$, to deduce most of the remaining cases. Of course S_1, $S_{1,3,5}$, and $S_{1,2,3,4,5}$ trivially have cost 4. That leaves only $S_{1,2,3,4}(x_1,x_2,x_3,x_4,x_5) = (x_1 \oplus x_2) \vee (x_2 \oplus x_3) \vee (x_3 \oplus x_4) \vee (x_4 \oplus x_5)$, which is discussed for general n in exercise 79.

24. As noted in the text, this conjecture holds for $n \le 5$.

25. It is $2^{2^n-1} - n - 1$, the number of nontrivial normal functions. (In any normal chain of length r that doesn't include all of these functions, $x_j \circ x_k$ will be a new function for some j and k in the range $1 \le j,k \le n+r$ and some normal binary operator $\circ$; so we can compute a new function with every new step, until we've got them all.)

26. False. For example, if $g = S_{1,3}(x_1,x_2,x_3)$ and $h = S_{2,3}(x_1,x_2,x_3)$, then $C(gh) = 5$ is the cost of a full adder; but $f = S_{2,3}(x_0,x_1,x_2,x_3)$ has cost 6 by Fig. 9.

27. Yes: The operations '$x_2 \leftarrow x_2 \oplus x_1$, $x_1 \leftarrow x_1 \oplus x_3$, $x_1 \leftarrow x_1 \wedge \bar{x}_2$, $x_1 \leftarrow x_1 \oplus x_3$, $x_2 \leftarrow x_2 \oplus x_3$' transform (x_1,x_2,x_3) into (z_1,z_0,x_3).

28. Let $v' = v'' = v \oplus (x \oplus y)$; $u' = ((v \oplus y) \overline{\subset} (x \oplus y)) \oplus u$, $u'' = ((v \oplus y) \vee (x \oplus y)) \oplus u$. Thus we can set $u_0 = 0$, $v_0 = x_1$, $u_j = ((v_{j-1} \oplus x_{2j+1}) \vee (x_{2j} \oplus x_{2j+1})) \oplus u_{j-1}$ if j is odd, $u_j = ((v_{j-1} \oplus x_{2j+1}) \overline{\subset} (x_{2j} \oplus x_{2j+1})) \oplus u_{j-1}$ if j is even, and $v_j = v_{j-1} \oplus (x_{2j} \oplus x_{2j+1})$, obtaining $(u_jv_j)_2 = (x_1 + \cdots + x_{2j+1}) \bmod 4$ for $1 \le j \le \lfloor n/2 \rfloor$. Set $x_{n+1} = 0$ if n is even. The function $[(x_1 + \cdots + x_n) \bmod 4 = 0] = \bar{u}_{\lfloor n/2 \rfloor} \wedge \bar{v}_{\lfloor n/2 \rfloor}$ is thereby computed in $\lfloor 5n/2 \rfloor - 2$ steps.

This construction is due to L. J. Stockmeyer, who proved that it is nearly optimal. In fact, the result of exercise 80 together with Figs. 9 and 10 shows that it is at most one step longer than a best possible chain, for all $n \geq 5$.

Incidentally, the analogous formula $u''' = ((v \oplus y) \wedge (x \oplus y)) \oplus u$ yields $(u'''v')_2 = ((uv)_2 + x - y) \bmod 4$. The simpler-looking function $((uv)_2 + x + y) \bmod 4$ costs 6, not 5.

29. To get an upper bound, assume that each full adder or half adder increases the depth by 3. If there are a_{jd} bits of weight 2^j and depth $3d$, we schedule at most $\lceil a_{jd}/3 \rceil$ subsequent bits of weights $\{2^j, 2^{j+1}\}$ and depth $3(d+1)$. It follows by induction that $a_{jd} \leq \binom{d}{j} 3^{-d} n + 4$. Hence $a_{jd} \leq 5$ when $d \geq \log_{3/2} n$, and the overall depth is at most $3 \log_{3/2} n + 3$. (Curiously, the actual depth turns out to be exactly 100 when $n = 10^7$.)

30. As usual, let νn denote the sideways addition of the bits in the binary representation of n itself. Then $s(n) = 5n - 2\nu n - 3\lfloor \lg n \rfloor - 3$.

31. After sideways addition in $s(n) < 5n$ steps, an arbitrary function of $(z_{\lfloor \lg n \rfloor}, \ldots, z_0)$ can be evaluated in $\sim 2n/\lg n$ steps at most, by Theorem L. [See O. B. Lupanov, *Doklady Akademii Nauk SSSR* **140** (1961), 322–325.]

32. Bootstrap: First prove by induction on n that $t(n) \leq 2^{n+1}$.

33. False, on a technicality: If, say, $N = \sqrt{n}$, at least n steps are needed. A correct asymptotic formula $N + O(\sqrt{N}\,) + O(n)$ can, however, be proved by first noting that the text's method gives $N + O(\sqrt{N}\,)$ when $N \geq 2^{n-1}$; otherwise, if $\lfloor \lg N \rfloor = n - k - 1$, we can use $O(n)$ operations to AND the quantity $\bar{x}_1 \wedge \cdots \wedge \bar{x}_k$ to the other variables $x_{k+1}, \ldots, x_n$, then proceed with n reduced by k.

(One consequence is that we can compute the symmetric functions $\{S_1, S_2, \ldots, S_n\}$ with cost $s(n) + n + O(\sqrt{n}\,) = 6n + O(\sqrt{n}\,)$ and depth $O(\log n)$.)

34. Say that an *extended* priority encoder has $n + 1 = 2^m$ inputs $x_0 x_1 \ldots x_n$ and $m + 1$ outputs $y_0 y_1 \ldots y_m$, where $y_0 = x_0 \vee x_1 \vee \cdots \vee x_n$. If Q'_m and Q''_m are extended encoders for $x'_0 \ldots x'_n$ and $x''_0 \ldots x''_n$, then Q_{m+1} works for $x'_0 \ldots x'_n x''_0 \ldots x''_n$ if we define $y_0 = y'_0 \vee y''_0$, $y_1 = y''_0$, $y_2 = y_1 ?\ y''_1 \colon y'_1$, $\ldots$, $y_{m+1} = y_1 ?\ y''_m \colon y'_m$. If P'_m is an ordinary priority encoder for $x'_1 \ldots x'_n$, we get P_{m+1} for $x'_1 \ldots x'_n x''_0 \ldots x''_n$ in a similar way.

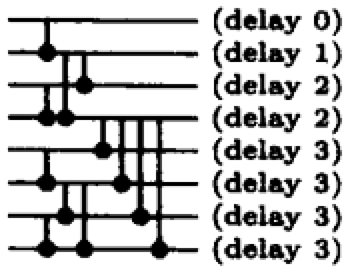
Starting with $m = 2$ and $y_2 = x_3 \vee (x_1 \wedge \bar{x}_2)$, $y_1 = x_2 \vee x_3$, $y_0 = x_0 \vee x_1 \vee y_1$, this construction yields P_m and Q_m of costs p_m and q_m, where $p_2 = 3$, $q_2 = 5$, and $p_{m+1} = 3m + p_m + q_m$, $q_{m+1} = 3m + 1 + 2q_m$ for $m \geq 2$. Consequently $p_m = q_m - m$ and $q_m = 15 \cdot 2^{m-2} - 3m - 4 \approx 3.75n$.

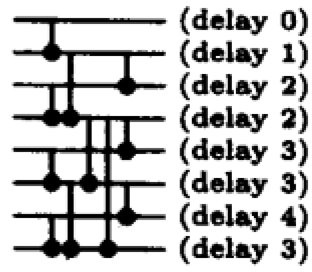
35. If $n = 2m$, compute $x_1 \wedge x_2, \ldots, x_{n-1} \wedge x_n$, then recursively form $x_1 \wedge \cdots \wedge x_{2k-2} \wedge x_{2k+1} \wedge \cdots \wedge x_n$ for $1 \leq k \leq m$, and finish in n more steps. If $n = 2m - 1$, use this chain for $n + 1$ elements; three steps can be eliminated by setting $x_{n+1} \leftarrow 1$. [I. Wegener, *The Complexity of Boolean Functions* (1987), exercise 3.25. The same idea can be used with *any* associative and commutative operator in place of $\wedge$.]

36. Recursively construct $P_n(x_1, \ldots, x_n)$ and $Q_n(x_1, \ldots, x_n)$ as follows, where P_n has optimum depth and Q_n has depth $\leq \lceil \lg n \rceil + 1$: The case $n = 1$ is trivial; otherwise P_n is obtained from $Q'_r(x_1, \ldots, x_r)$ and $P''_s(x_{r+1}, \ldots, x_n)$, where $r = \lceil n/2 \rceil$ and $s = \lfloor n/2 \rfloor$, by setting $y_j = y'_j$ for $1 \leq j \leq r$, $y_j = y'_r \wedge y''_{j-r}$ for $r < j \leq n$. And Q_n is obtained from either $P'_r(x_1 \wedge x_2, \ldots, x_{n-1} \wedge x_n)$ or $P'_r(x_1 \wedge x_2, \ldots, x_{n-2} \wedge x_{n-1}, x_n)$ by setting $y_1 = x_1$, $y_{2j} = y'_j$, $y_{2j+1} = y'_j \wedge x_{2j+1}$ for $1 \leq j < s$, and $y_{2s} = y'_s$, $y_n = y'_r$.

To prove validity we must show also that output y_n of Q_n has depth $\lceil \lg n \rceil$; notice that Q_{2m+1} would fail if we began it with $P'_m(x_1 \wedge x_2, \ldots, x_{2m-1} \wedge x_{2m})$ instead of with $P'_{m+1}(x_1 \wedge x_2, \ldots, x_{2m-1} \wedge x_{2m}, x_{2m+1})$, *except* when m is a power of 2.

These calculations can be performed in *minimum memory*, setting $x_{k(i)} \leftarrow x_{j(i)} \wedge x_{k(i)}$ at step i for some indices $j(i) < k(i)$. Thus we can illustrate the construction with diagrams analogous to the diagrams for sorting networks. For example,

$P_8 =$ 

; $Q_8 =$ 

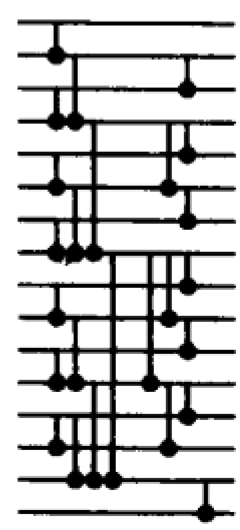
.

The costs p_n and q_n satisfy $p_n = \lfloor n/2 \rfloor + q_{\lceil n/2 \rceil} + p_{\lfloor n/2 \rfloor}$, $q_n = 2\lfloor n/2 \rfloor - 1 + p_{\lceil n/2 \rceil}$ when $n > 1$; for example, $(p_1, \ldots, p_7) = (q_1, \ldots, q_7) = (0, 1, 2, 4, 5, 7, 9)$. Setting $\bar{p}_n = 4n - p_n$ and $\bar{q}_n = 3n - q_n$ leads to simpler formulas, which prove that $p_n < 4n$ and $q_n < 3n$: $\bar{q}_n = \bar{p}_{\lceil n/2 \rceil} + [n \text{ even}]$; $\bar{p}_{4n} = \bar{p}_{2n} + \bar{p}_n + 1$, $\bar{p}_{4n+1} = \bar{p}_{2n} + \bar{p}_{n+1} + 1$, $\bar{p}_{4n+2} = \bar{p}_{2n+1} + \bar{p}_{n+1}$, $\bar{p}_{4n+3} = \bar{p}_{4n+2} + 2$. In particular, $1 + \bar{p}_{2^m} = F_{m+5}$ is a Fibonacci number.

[See *JACM* **27** (1980), 831–834. Slightly better chains are obtained if we use the otherwise-forbidden $P'_{\lfloor n/2 \rfloor}$ construction for Q_n when $n = 2^m + 1$, if we replace P_5 and P_6 by Q_5 and Q_6, and if we then replace $(P_9, P_{10}, P_{11}, P_{17})$ by $(Q_9, Q_{10}, Q_{11}, Q_{17})$.]

Notice that this construction works in general if we replace '$\wedge$' by *any* associative operator. In particular, the sequence of prefixes $x_1 \oplus \cdots \oplus x_k$ for $1 \le k \le n$ defines the conversion from Gray binary code to radix-2 integers, Eq. 7.2.1.1–(10).

37. The case $m = 15$, $n = 16$ is illustrated at the right.



(a) Let $x_{i..j}$ denote the original value of $x_i \wedge \cdots \wedge x_j$. Whenever the algorithm sets $x_k \leftarrow x_j \wedge x_k$, one can show that the previous value of x_k was $x_{j+1..k}$. After step S1, x_k is $x_{f(k)+1..k}$ where $f(k) = k \,\&\, (k-1)$ for $1 \le k < m$ and $f(m) = 0$. After step S2, x_k is $x_{1..k}$ for $1 \le k \le m$.

(b) The cost of S1 is $m - 1$, the cost of S2 is $m - 1 - \lceil \lg m \rceil$, and the cost of S3 is $n - m$. The final delay of x_k is $\lfloor \lg k \rfloor + \nu k - 1$ for $1 \le k < m$, and it is $\lceil \lg m \rceil + k - m$ for $m \le k \le n$. So the maximum delay for $\{x_1, \ldots, x_{m-1}\}$ turns out to be $g(m) = m - 1$ for $m < 4$, $g(m) = \lfloor \lg m \rfloor + \lfloor \lg \frac{m}{3} \rfloor$ for $m \ge 4$. We have $c(m,n) = m + n - 2 - \lceil \lg m \rceil$, $d(m,n) = \max(g(m), \lceil \lg m \rceil + n - m)$. Hence $c(m,n) + d(m,n) = 2n - 2$ whenever $n \ge m + g(m) - \lceil \lg m \rceil$.

(c) A table of values reveals that $d(n) = \lceil \lg n \rceil$ for $n < 8$, and $d(n) = \lfloor \lg(n - \lfloor \lg n \rfloor + 3) \rfloor + \lfloor \lg \frac{2}{3}(n - \lfloor \lg n \rfloor + 3) \rfloor - 1$ for $n \ge 8$. Stating this another way, we have $d(n) > d(n-1) > 0$ if and only if $n = 2^k + k - 3$ or $2^k + 2^{k-1} + k - 3$ for some $k > 1$. The minimum occurs for $m = n$ when $n < 8$; otherwise it occurs for $m = n - \lfloor \frac{2}{3}(n - \lfloor \lg n \rfloor + 3) \rfloor + 2 - [n = 2^k + k - 3 \text{ for some } k]$.

(d) Set $m \leftarrow m(n,d)$, where $m(n,d(n))$ is defined in the previous sentence and $m(n,d) = m(n-1,d-1)$ when $d > d(n)$. [See *J. Algorithms* **7** (1986), 185–201.]

38. (a) From top to bottom, $f_k(x_1, \ldots, x_n)$ is an elementary symmetric function also called the threshold function $S_{\ge k}(x_1, \ldots, x_n)$. (See exercise 5.3.4–28, Eq. 7.1.1–(90).)

(b) After calculating $\{S_1, \ldots, S_n\}$ in $\approx 6n$ steps as in answer 33, we can apply the method of exercise 37 to finish in $2n$ further steps.

But it is more interesting to design a Boolean chain specifically for the computation of the $2^m + 1$ threshold functions $g_k(x_1, \ldots, x_m) = [(x_1 \ldots x_m)_2 \ge k]$ for $0 \le k \le 2^m$. Since $[(x'x'')_2 \ge (y'y'')_2] = [(x')_2 \ge (y')_2 + 1] \vee ([(x')_2 \ge (y')_2] \wedge [(x'')_2 \ge (y'')_2])$, a divide-and-conquer construction analogous to a binary decoder solves this problem with a cost at most $2t(m)$.

Furthermore, if $2^{m-1} \le n < 2^m$, the cost $u(n)$ of computing $\{g_1, \ldots, g_n\}$ by this method turns out to be $2n + O(\sqrt{n})$, and it is quite reasonable when n is small:

n =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$u(n)$ =	0	1	2	4	7	7	8	12	15	17	19	19	20	21	22	27	32	34	36	36

Starting with sideways addition, we can sort n Boolean values in $s(n) + u(n) \approx 7n$ steps. A sorting network, which costs $2\hat{S}(n)$, is better when $n = 4$ but loses when $n \ge 8$. [See 5.3.4–(11); D. E. Muller and F. P. Preparata, *JACM* **22** (1975), 195–201.]

39. [*IEEE Transactions* **C-29** (1980), 737–738.] The identity

$$M_{r+s}(x_1, \ldots, x_r, x_{r+1}, \ldots, x_{r+s}; y_0, \ldots, y_{2^{r+s}-1}) = M_r(x_1, \ldots, x_r; y'_0, \ldots, y'_{2^r-1}),$$

where $y'_j = \bigvee_{k=0}^{2^s-1}(d_k \wedge y_{2^s j+k})$ and d_k is the kth output of an s-to-2^s decoder applied to $(x_{r+1}, \ldots, x_{r+s})$, shows that $C(M_{r+s}) \le C(M_r) + 2^{r+s} + 2^r(2^s-1) + t(s)$, where $t(s)$ is the cost (30) of the decoder. The depth is $D(M_{r+s}) = \max(D_x(M_{r+s}), D_y(M_{r+s}))$, where D_x and D_y denote the maximum depth of the x and y variables; we have $D_x(M_{r+s}) \le \max(D_x(M_r), 1 + s + \lceil \lg s \rceil + D_y(M_r))$ and $D_y(M_{r+s}) \le 1 + s + D_y(M_r)$.

Taking $r = \lceil m/2 \rceil$ and $s = \lfloor m/2 \rfloor$ yields $C(M_m) \le 2^{m+1} + O(2^{m/2})$, $D_y(M_m) \le m + 1 + \lceil \lg m \rceil$, and $D_x(M_m) \le D_y(M_m) + \lceil \lg m \rceil$.

40. We can, for example, let $f_{nk}(x) = \bigvee_{j=1}^{n+1-k}(l_j(x) \wedge r_{j+k-1}(x))$, where

$$l_j(x) = \begin{cases} x_j, & \text{if } j \bmod k = 0, \\ x_j \wedge l_{j+1}(x), & \text{if } j \bmod k \ne 0, \end{cases} \quad \text{for } 1 \le j \le n - (n \bmod k);$$

$$r_j(x) = \begin{cases} 1, & \text{if } j \bmod k = 0, \\ x_j \wedge r_{j-1}(x), & \text{if } j \bmod k \ne 0, \end{cases} \quad \text{for } k \le j \le n.$$

The cost is $4n - 3k - 3\lfloor \frac{n}{k} \rfloor - \lfloor \frac{n-1}{k} \rfloor + 2 - (n \bmod k)$.

A recursive solution is preferable when n is small or k is small: Observe that

$$f_{nk}(x) = \begin{cases} x_{n-k+1} \wedge \cdots \wedge x_k \wedge \\ \quad f_{(2n-2k)(n-k)}(x_1, \ldots, x_{n-k}, x_{k+1}, \ldots, x_n), & \text{for } k < n < 2k; \\ f_{\lfloor (n+k)/2 \rfloor k}(x_1, \ldots, x_{\lfloor (n+k)/2 \rfloor}) \vee \\ \quad f_{\lfloor (n+k-1)/2 \rfloor k}(x_{\lfloor (n-k)/2 \rfloor + 1}, \ldots, x_n), & \text{for } n \ge 2k. \end{cases}$$

The cost of this solution can be shown to equal $n - 1 + \sum_{j=1}^{n-k} \lfloor \lg j \rfloor$ when $k \le n < 2k$, and it lies asymptotically between $(m + \alpha_k - 1)n + O(km)$ and $(m + 2 - 2/\alpha_k)n + O(km)$ as $n \to \infty$, where $m = \lfloor \lg k \rfloor$ and $1 < \alpha_k = (k+1)/2^m \le 2$.

A marriage of these methods is better yet; the optimum cost is unknown.

41. Let $c(m)$ be the cost of computing both $(x)_2 + (y)_2$ and $(x)_2 + (y)_2 + 1$ by the conditional-sum method when x and y have $n = 2^m$ bits, and let $c'(m)$ be the cost of the simpler problem of computing just $(x)_2 + (y)_2$. Then $c(m+1) = 2c(m) + 6 \cdot 2^m + 2$, $c'(m+1) = c(m) + c'(m) + 3 \cdot 2^m + 1$. (Bit z_n of the sum costs 1; but bits z_k for $n < k \le 2n + 1$ cost 3, because they have the form $c?\ a_k\colon b_k$ where c is a carry bit.) If we start with $n = 1$ and $c(0) = 3$, $c'(0) = 2$, the solution is $c(m) = (3m + 5)2^m - 2$, $c'(m) = (3m + 2)2^m - m$. But improved constructions for the case $n = 2$ allow us to start with $c(1) = 11$ and $c'(1) = 7$; then the solution is $c(m) = (3m + \frac{7}{2})2^m - 2$, $c'(m) = (3m + \frac{1}{2})2^m - m + 1$. In either case the depth is $2m + 1$. [See J. Sklansky, *IRE Transactions* **EC-9** (1960), 226–231.]

42. (a) Since $\langle x_k y_k c_k \rangle = u_k \vee (v_k \wedge c_k)$, we can use (26) and induction.

(b) Notice that $U_k^{k+1} = u_k$ and $V_k^{k+1} = v_k$; use induction on $j - i$. [See A. Weinberger and J. L. Smith, *IRE Transactions* **EC-5** (1956), 65–73; R. P. Brent and H. T. Kung, *IEEE Transactions* **C-31** (1982), 260–264.]

(c) First, for $l = 1, 2, \ldots, m-1$, and for $1 \le k \le n$, compute V_i^k for all multiples i of $h(l)$ in the range $k_l \ge i \ge k_{l+1}$, where $k_l = h(l)\lfloor (k-1)/h(l) \rfloor$ denotes the largest multiple of $h(l)$ that is less than k. For example, when $l = 2$ and $k = 99$, we compute V_{96}^{99}, $V_{88}^{99} = V_{96}^{99} \wedge V_{88}^{96}$, $V_{80}^{99} = V_{88}^{99} \wedge V_{80}^{88}$, $\ldots$, $V_{64}^{99} = V_{72}^{99} \wedge V_{64}^{72}$; this is a prefix computation using the values V_{96}^{99}, V_{88}^{96}, V_{80}^{88}, $\ldots$, V_{64}^{72} that were computed when $l = 2$. Using the method of exercise 36, step l adds at most l levels to the depth, and it requires a total of $(p_1 + p_2 + \cdots + p_{2^l})n/2^l = O(2^l n)$ gates.

Then, again for $l = 1, 2, \ldots, m-1$, and for $1 \le k \le n$, compute U_i^k for $i = k_{l+1}$, using the "unrolled" formula

$$U_{k_{l+1}}^k = U_{k_l}^k \vee \bigvee_{\substack{k_l > j \ge k_{l+1} \\ h(l) \backslash j}} (V_{j+h(l)}^k \wedge U_j^{j+h(l)}).$$

For example, the unrolled formula when $l = 3$ and $k = 99$ is

$$U_{64}^{99} = U_{96}^{99} \vee (V_{96}^{99} \wedge U_{88}^{96}) \vee (V_{88}^{99} \wedge U_{80}^{88}) \vee (V_{80}^{99} \wedge U_{72}^{80}) \vee (V_{72}^{99} \wedge U_{64}^{72}).$$

Every such U_i^k is a union of at most 2^l terms, so it can be computed with depth $\le l$ in addition to the depth of each term. The total cost of this phase for $1 \le k \le n$ is $(0 + 2 + 4 + \cdots + (2^l - 2))n/2^l = O(2^l n)$.

The overall cost to compute all necessary U's and V's is therefore $\sum_{l=1}^{m-1} O(2^l n) = O(2^m n)$. (Furthermore the quantities V_0^k aren't actually needed, so we save the cost of $\sum_{l=1}^{m-1} h(l) p_{2^l}$ gates.) For example, when $m = (2, 3, 4, 5)$ we obtain Boolean chains for the addition of $(2, 8, 64, 1024)$-bit numbers, respectively, with overall depths $(3, 7, 11, 16)$ and costs $(7, 64, 1254, 48470)$.

[This construction is due to V. M. Khrapchenko, *Problemy Kibernetiki* **19** (1967), 107–122, who also showed how to combine it with other methods so that the overall cost will be $O(n)$ while still achieving depth $\lg n + O(\sqrt{\log n})$. However, his combined method is purely of theoretical interest, because it requires $n > 2^{64}$ before the depth becomes less than $2 \lg n$. Another way to achieve small depth using the recurrences in (b) can be based on the Fibonacci numbers: The Fibonacci method computes the carries with depth $\log_\phi n + O(1) \approx 1.44 \lg n$ and cost $O(n \log n)$. For example, it yields chains for binary addition with the following characteristics:

$n =$	4	8	16	32	64	128	256	512	1024
depth	6	7	9	10	12	13	15	16	18
cost	24	71	186	467	1125	2648	6102	13775	30861

See D. E. Knuth, *The Stanford GraphBase* (1994), 276–279.

Charles Babbage found an ingenious mechanical solution to the analogous problem for addition in radix 10, claiming that his design would be able to add numbers of arbitrary precision in constant time; for this to work he would have needed idealized, rigid components with vanishing clearances. See H. P. Babbage, *Babbage's Calculating Engines* (1889), 334–335. Curiously, an equivalent idea works fine with physical transistors, although it cannot be expressed in terms of Boolean chains; see P. M. Fenwick, *Comp. J.* **30** (1987), 77–79.]

43. (a) Let $A = B = Q = \{0,1\}$ and $q_0 = 0$. Define $c(q,a) = d(q,a) = \bar{q} \wedge a$.

(b) The key idea is to construct the functions $d_1(q) \ldots d_{n-1}(q)$, where $d_1(q) = d(q,a_1)$ and $d_{j+1}(q) = d(d_j(q), a_j)$. In other words, $d_1 = d^{(a_1)}$ and $d_{j+1} = d_j \circ d^{(a_j)}$, where $d^{(a)}$ is the function that takes $q \mapsto d(q,a)$ and where $\circ$ denotes composition of functions. Each function d_j can be encoded in binary notation, and $\circ$ is an associative operation on these binary representations. Hence the functions $d_1 d_2 \ldots d_{n-1}$ are the prefixes $d^{(a_1)}$, $d^{(a_1)} \circ d^{(a_2)}$, $\ldots$, $d^{(a_1)} \circ \cdots \circ d^{(a_{n-1})}$; and $q_1 q_2 \ldots q_n = q_0 d_1(q_0) \ldots d_{n-1}(q_0)$.

(c) Represent a function $f(q)$ by its truth table $f_0 f_1$. Then the composition $f_0 f_1 \circ g_0 g_1$ is $h_0 h_1$, where the functions $h_0 = f_0?\ g_1\colon g_0$ and $h_1 = f_1?\ g_1\colon g_0$ are muxes that can each be computed with cost 3 and depth 2. (The combined cost $C(h_0 h_1)$ is only 5, but we are trying to keep the depth small.) The truth table for $d^{(a)}$ is $a0$. Using exercise 36, we can therefore compute the truth tables $d_{10} d_{11} d_{20} d_{21} \ldots d_{(n-1)0} d_{(n-1)1}$ with cost $\le 6p_{n-1} < 24n$ and depth $\le 2\lceil \lg(n-1) \rceil$; then $b_j = \bar{q}_j \wedge a_j = \bar{d}_{(j-1)0} \wedge a_j$. (These cost estimates are quite conservative; substantial simplifications arise because of the 0s in the initial truth tables of $d^{(a_j)}$ and because many of the intermediate values d_{j1} are never used. For example, when $n = 5$ the actual cost is only 10, not $6p_4 + 4 = 28$; the actual depth is 4, not $1 + 2\lceil \lg 4 \rceil = 5$.)

44. The inputs may be regarded as the string $x_0 y_0 x_1 y_1 \ldots x_{n-1} y_{n-1}$ whose elements belong to the four-letter alphabet $A = \{00, 01, 10, 11\}$; there are two states $Q = \{0,1\}$, representing a possible carry bit, with $q_0 = 0$; the output alphabet is $B = \{0,1\}$; and we have $c(q,xy) = q \oplus x \oplus y$, $d(q,xy) = \langle qxy \rangle$. In this case, therefore, the finite state transducer is essentially described by a full adder.

Only three of the four possible functions of q occur when we compose the mappings $d^{(xy)}$. We can encode them as $u \vee (q \wedge v)$. The initial functions $d^{(xy)}$ have $u = x \wedge y$, $v = x \oplus y$; and the composition $(uv) \circ (u'v')$ is $u''v''$, where $u'' = u' \vee (v' \wedge u)$ and $v'' = v \wedge v'$.

When $n = 4$, for example, the chain has the following form, using the notation of exercise 42: $U_k^{k+1} = x_k \wedge y_k$, $V_k^{k+1} = x_k \oplus y_k$, for $0 \le k < 4$; $U_0^2 = U_1^2 \vee (V_1^2 \wedge U_0^1)$, $U_2^4 = U_3^4 \vee (V_3^4 \wedge U_2^3)$, $V_2^4 = V_2^3 \wedge V_3^4$; $U_0^3 = U_2^3 \vee (V_2^3 \wedge U_0^2)$, $U_0^4 = U_2^4 \vee (V_2^4 \wedge U_0^2)$; $z_0 = V_0^1$, $z_1 = U_0^1 \oplus V_1^2$, $z_2 = U_0^2 \oplus V_2^3$, $z_3 = U_0^3 \oplus V_3^4$, $z_4 = U_0^4$. The total cost is 20, and the maximum depth is 5.

In general the cost will be $2n + 3p_n$ in the notation of exercise 36, because we need $2n$ gates for the initial u's and v's, then $3p_n$ gates for the prefix computation; the $n-1$ additional gates needed to form z_j for $0 < j < n$ are compensated by the fact that we need not compute V_0^j for $1 < j \le n$. Therefore the total cost is $14 \cdot 2^m - 3F_{m+5} + 3$, clearly superior to the conditional-sum method (which has the same depth $2m + 1$):

$n =$	2	4	8	16	32	64	128	256	512	1024
cost of conditional-sum chain	7	25	74	197	492	1179	2746	6265	14072	31223
cost of Ladner–Fischer chain	7	20	52	125	286	632	1363	2888	6040	12509

[George Boole introduced his Algebra in order to show that logic can be understood in terms of arithmetic. Eventually logic became so well understood, the situation was reversed: People like Shannon and Zuse began in the 1930s to design circuits for arithmetic in terms of logic, and since then many approaches to the problem of parallel addition have been discovered. The first Boolean chains of cost $O(n)$ and depth $O(\log n)$ were devised by Yu. P. Ofman, *Doklady Akademii Nauk SSSR* **145** (1962), 48–51. His chains were similar to the construction above, but the depth was approximately $4m$.]

45. That argument would indeed be simpler, but it wouldn't be strong enough to prove the desired result. (Many chains with steps of fan-out 0 inflate the simpler estimate.)

The text's permutation-enhanced proof technique was introduced by J. E. Savage in his book *The Complexity of Computing* (New York: Wiley, 1976), Theorem 3.4.1.

46. When $r = 2^n/n + O(1)$ we have $\ln(2^{2r+1}(n+r-1)^{2r}/(r-1)!) = r\ln r + (1+\ln 4)r + O(n) = (2^n/n)(n\ln 2 - \ln n + 1 + \ln 4) + O(n)$. So $\alpha(n) \le (n/(4e))^{-2^n/n+O(n/\log n)}$, which approaches zero quite rapidly indeed when $n > 4e$.

(In fact, (32) gives $\alpha(11) < 7.6 \times 10^7$, $\alpha(12) < 4.2 \times 10^{-6}$, $\alpha(13) < 1.2 \times 10^{-38}$.)

47. Restrict permutations to the $(r-m)!$ cases where $i\pi = i$ for $1 \le i \le n$ and $(n+r+1-k)\pi$ is the kth output. Then we get $(r-m)!\,c(m,n,r) \le 2^{2r+1}(n+r-1)^{2r}$ in place of (32). Hence, as in exercise 46, almost all such functions have cost exceeding $2^n m/(n+\lg m)$ when $m = O(2^n/n^2)$.

48. (a) Not surprisingly, this lower bound on $C(n)$ is rather crude when n is small:

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r(n) =$	1	1	2	3	5	9	16	29	54	99	184	343	639	1196	2246	4229

(b) The bootstrap method (see *Concrete Mathematics* §9.4) yields

$$r(n) = \frac{2^n}{n}\Bigl(1 + \frac{\lg n - 2 - 1/\ln 2}{n} + O\Bigl(\frac{\log n}{n^2}\Bigr)\Bigr).$$

49. The number of normal Boolean functions that can be represented by a formula of length $\le r$ is at most $5^r n^{r+1} g_r$, where g_r is the number of oriented binary trees with r internal nodes. Set $r = 2^n/\lg n - 2^{n+2}/(\lg n)^2$ in this formula and divide by 2^{2^n-1} to get an upper bound on the fraction of functions with $L(f) \le r$. The result rapidly approaches zero, by exercise 2.3.4.4–7, because it is $O((5\alpha/16)^{2^n/\lg n})$ where $\alpha \approx 2.483$.

[J. Riordan and C. E. Shannon obtained a similar lower bound for series-parallel switching networks in *J. Math. and Physics* **21** (1942), 83–93; such networks are equivalent to formulas in which only canalizing operators are used. R. E. Krichevsky obtained more general results in *Problemy Kibernetiki* **2** (1959), 123–138, and O. B. Lupanov gave an asymptotically matching upper bound in *Prob. Kibernetiki* **3** (1960), 61–80.]

50. (a) Using subcube notation as in exercise 7.1.1–30, the prime implicants are 00001*, (0001*1), 0100*1, 0111*1, 1010*1, 101*11, 00*011, 00*101, (01*111), 11*101, (0*1101), (1*0101), 1*1011, 0*0*11, *00101, (*01011), (*11101), where the parenthesized subcubes are omitted in a shortest DNF. (b) Similarly, the prime clauses and a shortest CNF are given by 00111*, 01010*, 10110*, 0110**, 00*00*, 11*00*, 11*11*, (0*100*), (1*00**), 1*0*1*, (1****0), *0000*, (*1100*), *1***0, **1**0, ***1*0, and (****00). (Thus the CNF is $(x_1 \vee x_2 \vee \bar x_3 \vee \bar x_4 \vee \bar x_5) \wedge (x_1 \vee \bar x_2 \vee x_3 \vee \bar x_4 \vee x_5) \wedge \cdots \wedge (\bar x_4 \vee x_6)$.)

51. $f = ([x_5x_6 \in \{01\}] \wedge [(x_1x_2x_3x_4)_2 \in \{1,3,4,7,9,10,13,15\}]) \vee ([x_5x_6 \in \{10,11\}] \wedge [x_1x_2x_3x_4 = 0000]) \vee ([x_5x_6 \in \{11\}] \wedge [(x_1x_2x_3x_4)_2 \in \{1,2,4,5,7,10,11,14\}])$.

52. The small-n results are quite different from those that work asymptotically:

n	k	l	(38)	n	k	l	(38)	n	k	l	(38)	n	k	l	(38)
5	2	2	39	8	3	2	175	11	4	4	803	14	5	5	4045
6	2	2	67	9	3	2	279	12	4	3	1329	15	5	5	7141
7	2	1	109	10	4	4	471	13	5	6	2355	16	5	4	12431

(Optimizations like the fact that $[x_1x_2 \in \{00,01\}] = \bar x_1$ usually reduce the cost further.)

53. First note that $2^k/l \le n - 3\lg n$, hence $m_i \le n - 3\lg n + 1$ and $2^{m_i} = O(2^n/n^3)$. Also $l = O(n)$ and $t(n-k) = O(2^n/n^2)$. So (38) reduces to $l \cdot 2^{n-k} + O(2^n/n^2) = 2^n/(n - 3\lg n) + O(2^n/n^2)$.

54. The greedy-footprint heuristic gives a chain of length 15:

$$\begin{array}{lll}
x_5 = x_1 \oplus x_2, & x_{10} = x_4 \oplus x_5, & f_2 = x_{15} = \bar{x}_5 \wedge x_8, \\
x_6 = x_2 \oplus x_3, & x_{11} = x_7 \vee x_{10}, & f_3 = x_{16} = x_4 \wedge \bar{x}_{12}, \\
x_7 = x_1 \wedge \bar{x}_3, & x_{12} = x_6 \oplus x_{11}, & f_4 = x_{17} = x_1 \wedge x_8, \\
x_8 = x_4 \wedge \bar{x}_6, & x_{13} = \bar{x}_{10} \wedge x_{12}, & f_5 = x_{18} = x_7 \wedge x_9, \\
x_9 = x_4 \wedge \bar{x}_5, & f_1 = x_{14} = x_6 \wedge \bar{x}_{11}, & f_6 = x_{19} = \bar{x}_3 \wedge x_{13}.
\end{array}$$

The minterm-first method corresponds to a chain of length 22, after we remove steps that are never used:

$$\begin{array}{lll}
x_5 = \bar{x}_1 \wedge \bar{x}_2, & x_{13} = x_5 \wedge x_{10}, & x_{20} = x_8 \wedge x_{11}, \\
x_6 = \bar{x}_1 \wedge x_2, & x_{14} = x_5 \wedge x_{11}, & f_6 = x_{21} = x_{15} \vee x_{18}, \\
x_7 = x_1 \wedge \bar{x}_2, & x_{15} = x_6 \wedge x_9, & f_1 = x_{22} = x_{13} \vee x_{21}, \\
x_8 = x_1 \wedge x_2, & x_{16} = x_6 \wedge x_{11}, & f_2 = x_{23} = x_{12} \vee x_{20}, \\
x_9 = \bar{x}_3 \wedge x_4, & x_{17} = x_7 \wedge x_9, & x_{24} = x_{14} \vee x_{16}, \\
x_{10} = x_3 \wedge \bar{x}_4, & x_{18} = x_7 \wedge x_{11}, & f_3 = x_{25} = x_{24} \vee x_{19}, \\
x_{11} = x_3 \wedge x_4, & f_5 = x_{19} = x_8 \wedge x_9, & f_4 = x_{26} = x_{17} \vee x_{20}. \\
x_{12} = x_5 \wedge x_9, & &
\end{array}$$

(The distributive law could replace the computation of x_{14}, x_{16}, and x_{24} by two steps.)

Incidentally, the three functions in the answer to exercise 51 can be computed in only ten steps:

$$\begin{array}{lll}
x_5 = x_2 \vee x_4, & f_3 = x_9 = x_6 \oplus x_8, & x_{12} = x_2 \oplus x_3, \\
x_6 = \bar{x}_1 \wedge x_5, & x_{10} = x_1 \oplus x_8, & x_{13} = \bar{x}_{10} \wedge x_{12}, \\
x_7 = x_2 \wedge x_4, & \bar{f}_2 = x_{11} = x_9 \vee x_{10}, & f_1 = x_{14} = x_4 \oplus x_{13}. \\
x_8 = x_3 \wedge \bar{x}_7, & &
\end{array}$$

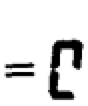
55. The optimum two-level DNF and CNF representations in answer 50 cost 53 and 43, respectively. Formula (37) costs 30, when optimized as in exercise 54. The alternative in exercise 51 costs only 17. But the catalog of optimum five-variable chains suggests

$$\begin{array}{llll}
x_7 = \bar{x}_1 \wedge x_2, & x_{11} = x_5 \wedge x_{10}, & x_{15} = x_{13} \oplus x_{14}, & x_{18} = \bar{x}_4 \wedge x_{17}, \\
x_8 = x_3 \oplus x_7, & x_{12} = x_5 \vee x_{10}, & x_{16} = x_5 \wedge \bar{x}_{10}, & x_{19} = x_6 \wedge x_{15}, \\
x_9 = x_2 \wedge x_8, & x_{13} = x_4 \wedge \bar{x}_{11}, & x_{17} = \bar{x}_3 \wedge x_{16}, & x_{20} = x_{18} \vee x_{19}, \\
x_{10} = x_1 \oplus x_9, & x_{14} = x_7 \wedge x_{12}, & &
\end{array}$$

for this six-variable function. Is there a better way?

56. If we care about at most two values, the function can be either constant or x_j or $\bar{x}_j$.

57. The truth tables for x_5 through x_{17}, in hexadecimal notation, are respectively 0ff0, 2222, 33cc, 0d0d, 7777, 5d5d, 3ec1, 6b94, 4914, 4804, 060b, 2020, 7007. So we get

1010 = , 1011 = , 1012 = , 1013 = , 1014 = , 1015 = .

58. The truth tables of all cost-7 functions with exactly eight 1s in their truth tables are equivalent to either 0779, 169b, or 179a. Combining these in all possible ways yields 9656 solutions that are distinct under permutation and/or complementation of $\{x_1, x_2, x_3, x_4\}$ as well as under permutation and/or complementation of $\{f_1, f_2, f_3, f_4\}$.

59. The greedy-footprint heuristic produces the following 17-step chain:

$$\begin{aligned}
x_5 &= x_1 \vee x_4, & x_{11} &= x_8 \vee x_9, & x_{17} &= \bar{x}_2 \wedge x_3,\\
x_6 &= x_1 \oplus x_3, & x_{12} &= x_1 \oplus x_{11}, & f_1 = x_{18} &= x_{13} \oplus x_{15},\\
x_7 &= x_2 \oplus x_4, & x_{13} &= x_5 \wedge \bar{x}_9, & f_2 = x_{19} &= x_{11} \wedge \bar{x}_{16},\\
x_8 &= \bar{x}_4 \wedge x_6, & x_{14} &= x_5 \wedge x_{12}, & f_3 = x_{20} &= x_{12} \oplus x_{17},\\
x_9 &= x_3 \oplus x_7, & x_{15} &= x_2 \wedge x_6, & f_4 = x_{21} &= x_{10} \wedge \bar{x}_{14}.\\
x_{10} &= x_2 \vee x_3, & x_{16} &= x_2 \wedge \bar{x}_6,
\end{aligned}$$

The initial functions all have large footprints, so we can't achieve $C(f_1 f_2 f_3 f_4) = 28$; but a slightly more difficult S-box probably does exist.

60. One way is $u_1 = x_1 \oplus y_1$, $u_2 = x_2 \oplus y_2$, $v_1 = y_2 \oplus u_1$, $v_2 = y_1 \oplus u_2$, $z_1 = v_1 \wedge \bar{u}_2$, $z_2 = v_2 \wedge \bar{u}_1$.

61. Viewing these partial functions of six variables as 4×16 truth tables, with rows governed by $x_1 y_1$, our knowledge of 4-bit functions suggests good ways to compute the rows and leads to the following 25-step solution: $t_1 = x_2 \wedge y_2$, $t_2 = x_3 \wedge y_3$, $t_3 = x_2 \vee y_2$, $t_4 = x_3 \vee y_3$, $t_5 = t_1 \oplus t_2$, $t_6 = t_1 \vee t_2$, $t_7 = t_4 \wedge \bar{t}_5$, $t_8 = t_3 \oplus t_6$, $t_9 = x_2 \oplus y_2$, $t_{10} = t_4 \oplus t_9$, $t_{11} = t_5 \wedge \bar{t}_{10}$, $t_{12} = t_3 \oplus t_4$, $t_{13} = x_1 \vee y_1$, $t_{14} = t_8 \oplus t_{12}$, $t_{15} = t_{13} \wedge \bar{t}_{14}$, $t_{16} = t_4 \oplus t_7$, $t_{17} = t_{13} \wedge \bar{t}_{16}$, $t_{18} = t_3 \vee t_4$, $t_{19} = x_1 \oplus y_1$, $t_{20} = t_{19} \wedge \bar{t}_{18}$, $t_{21} = t_8 \oplus t_{15}$, $t_{22} = t_7 \oplus t_{17}$, $z_1 = t_{11} \vee t_{20}$, $z_2 = t_{21} \wedge \bar{t}_{20}$, $z_3 = t_{22} \wedge \bar{t}_{20}$. (Is there a better way? Gilbert Lee has found a 17-step solution if the inputs are represented by 000, 001, 011, 101, and 111.)

62. There are $\binom{2^n}{2^n d} 2^{2^n c}$ such functions, at most $\binom{2^n}{2^n d} t(n,r)$ of which have cost $\le r$. So we can argue as in exercise 46 to conclude from (32) that the fraction with cost $\le r = \lfloor 2^n c/n \rfloor$ is at most $2^{2r+1-2^n c}(n+r-1)^{2r}/(r-1)! = 2^{-r \lg n + O(r)}$.

63. [*Problemy Kibernetiki* **21** (1969), 215–226.] Put the truth table in a $2^k \times 2^{n-k}$ array as in Lupanov's method, and suppose there are c_j cares in column j, for $0 \le j < 2^{n-k}$. Break that column into $\lfloor c_j/m \rfloor$ subcolumns that each have m cares, plus a possibly empty subcolumn at the bottom that contains fewer than m of them. The hint tells us that at most 2^{m+k} column vectors suffice to match the 0s and 1s of every subcolumn that has a specified top row i_0 and bottom row i_1. With $O(m 2^{m+3k})$ operations we can therefore construct $O(2^{m+3k})$ functions $g_t(x_1, \ldots, x_k)$ from the minterms of $\{x_1, \ldots, x_k\}$, so that every subcolumn matches some type t. And for every type t we can construct functions $h_t(x_{k+1}, \ldots, x_n)$ from the minterms of $\{x_{k+1}, \ldots, x_n\}$, specifying the columns that match t; the cost is at most $\sum_j (\lfloor c_j/m \rfloor + 1) \le 2^n c/m + 2^{n-k}$. Finally, $f = \bigvee_t (g_t \wedge h_t)$ requires $O(2^{m+3k})$ additional steps. Choosing $k = \lfloor 2 \lg n \rfloor$ and $m = \lceil n - 9 \lg n \rceil$ makes the total cost at most $(2^n c/n)(1 + 9n^{-1} \lg n + O(n^{-1}))$.

Of course we need to prove the hint, which is due to E. I. Nechiporuk [*Doklady Akad. Nauk SSSR* **163** (1965), 40–42]. In fact, $2^m(1 + \lceil k \ln 2 \rceil)$ vectors suffice (see S. K. Stein, *J. Combinatorial Theory* **A16** (1974), 391–397): If we choose $q = 2^m \lceil k \ln 2 \rceil$ vectors at random, not necessarily distinct, the expected number of untouched subcubes is $\binom{k}{m} 2^m (1 - 2^{-m})^q < \binom{k}{m} 2^m e^{-q 2^{-m}} < 2^m$. (An explicit construction would be nicer.)

For extensive generalizations — tolerating a percentage of errors and specifying the density of 1s — see N. Pippenger, *Mathematical Systems Theory* **10** (1977), 129–167.

64. It's exactly the game of tic-tac-toe, if we number the cells $\begin{smallmatrix} 6&1&8\\ 7&5&3\\ 2&9&4 \end{smallmatrix}$ as in an ancient Chinese magic square. [Berlekamp, Conway, and Guy use this numbering scheme to present a complete analysis of tic-tac-toe in their book *Winning Ways* **3** (2003), 732–736.]

65. One solution is to replace the "defending" moves d_j by "attacking" moves a_j and "counterattacking" moves c_j, and to include them only for corner cells $j \in \{1, 3, 9, 7\}$.

Let $j \cdot k = (jk) \bmod 10$; then

$$\begin{matrix} j\cdot 1 & j\cdot 2 & j\cdot 3 \\ j\cdot 4 & j\cdot 5 & j\cdot 6 \\ j\cdot 7 & j\cdot 8 & j\cdot 9 \end{matrix}$$

gives us another way to look at the tic-tac-toe diagram, when j is a corner, because $j \perp 10$. The precise definition of a_j and c_j is then

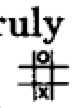
$$a_j = m_j \wedge \bigl((x_{j\cdot 3} \wedge \beta_{(j\cdot 8)(j\cdot 9)} \wedge (o_{j\cdot 4} \oplus o_{j\cdot 6})) \vee (x_{j\cdot 7} \wedge \beta_{(j\cdot 6)(j\cdot 9)} \wedge (o_{j\cdot 2} \oplus o_{j\cdot 8}))$$
$$\vee\, (m_{j\cdot 9} \wedge ((m_{j\cdot 8} \wedge x_{j\cdot 2} \wedge \overline{(o_{j\cdot 3} \oplus o_{j\cdot 6})}) \vee (m_{j\cdot 6} \wedge x_{j\cdot 4} \wedge \overline{(o_{j\cdot 7} \oplus o_{j\cdot 8})})))\bigr);$$
$$c_j = d_j \wedge \overline{(x_{j\cdot 6} \wedge o_{j\cdot 7})} \wedge \overline{(x_{j\cdot 8} \wedge o_{j\cdot 3})} \wedge \bar{d}_{j\cdot 9};$$

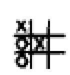
here $d_j = m_j \wedge \beta_{(j\cdot 2)(j\cdot 3)} \wedge \beta_{(j\cdot 4)(j\cdot 7)}$ takes the place of (51). We also define

$$u = (x_1 \oplus x_3) \oplus (x_7 \oplus x_9), \qquad v = (o_1 \oplus o_3) \oplus (o_7 \oplus o_9), \qquad t = m_2 \wedge m_6 \wedge m_8 \wedge m_4 \wedge (u \vee \bar{v}),$$
$$z_j = \begin{cases} m_j \wedge \bar{t}, & \text{if } j = 5, \\ m_j \wedge \bar{d}_{j\cdot 9}, & \text{if } j \in \{1,3,9,7\}, \\ m_j, & \text{if } j \in \{2,6,8,4\}, \end{cases}$$

in order to cover a few more exceptional cases. Finally the sequence of rank-ordered moves $d_5d_1d_3d_9d_7d_2d_6d_8d_4m_5m_1m_3m_9m_7m_2m_6m_8m_4$ in (53) is replaced by the sequence $a_1a_3a_9a_7c_1c_3c_9c_7z_5z_1z_3z_9z_7z_2z_6z_8z_4$; and we replace $(d_j \wedge \bar{d}'_j) \vee (m_j \wedge \bar{m}'_j)$ in (55) by $(a_j \wedge \bar{a}'_j) \vee (c_j \wedge \bar{c}'_j) \vee (z_j \wedge \bar{z}'_j)$ when j is a corner cell, otherwise simply by $(z_j \wedge \bar{z}'_j)$.

(Notice that this machine is required to move correctly from *all* legal positions, even when those positions couldn't arise after the machine had made X's earlier moves. We essentially allow humans to play the game until they ask the machine for advice. Otherwise great simplifications would be possible. For example, if X always goes first, it could grab the center cell and eliminate a huge number of future possibilities; fewer than $8 \times 6 \times 4 \times 2 = 384$ games could arise. Even if O goes first, there are fewer than $9 \times 7 \times 5 \times 3 = 945$ possible scenarios against a fixed strategy. In fact, the actual number of different games with the strategy defined here turns out to be $76 + 457$, of which $72 + 328$ are won by the machine and the rest belong to the cat.)

66. The Boolean chain in the previous answer fulfills its mission of making correct moves from all 4520 legal positions, where correctness was essentially defined to mean that the worst-case final outcome is maximized. But a truly great tic-tac-toe player would do things differently. For example, from position  the machine takes the center, , and O probably draws by playing in a corner. But moving to  would give O only two chances to avoid defeat. [See Martin Gardner, *Hexaflexagons and Other Mathematical Diversions*, Chapter 4.]

Furthermore the best move from a position like  is to  instead of winning immediately; then if the reply is , move to . That way you still win, but without humiliating your opponent so badly.

Finally, even the concept of a single "best move" is flawed, because a good player will choose different moves in different games (as Babbage observed).

> *It might be thought that programing a digital computer to play ticktacktoe, or designing special circuits for a ticktacktoe machine, would be simple. This is true unless your aim is to construct a master robot that will win the maximum number of games against inexperienced players.*
>
> — MARTIN GARDNER, *The Scientific American Book of Mathematical Puzzles & Diversions* (1959)

67. The author's best effort, with 1734 gates, was constructed by adapting the method of Sholomov in answer 63: First divide the truth tables into 64 rows for $o_5x_5o_2o_6o_8o_4$ and 4096 columns for the other 12 input variables. Then place appropriate 1s into "care" positions, in such a way that the columns have relatively few 1s. Then find a small number of column types that match the cares in all columns; 23 types suffice for the c function, 20 types for s, and 6 for m. We can then compute each output as $\bigvee(g_t \wedge h_t)$, sharing much of the work of the minterm calculations within g_t and h_t.

[This exercise was inspired by a discussion in John Wakerly's book *Digital Design* (Prentice–Hall, 3rd edition, 2000), §6.2.7. Incidentally, Babbage planned to choose among k possible moves by looking at $N \bmod k$, where N was the number of games won so far; he didn't realize that successive moves would tend to be highly correlated until N changed. Much better would have been to let N be the number of *moves made* so far.]

68. No. That method yields a "uniform" chain with a comprehensible structure, but its cost is 2^n times a polynomial in n. A circuit with approximately $2^n/n$ gates, constructed by Theorem L, exists but is more difficult to fabricate. (Incidentally, $C(\pi_5) = 10$.)

69. (a) One can, for example, verify this result by trying all 64 cases.

(b) If x_m lies in the same row or column as x_i, and also in the same row or column as x_j, we have $\alpha_{111} = \alpha_{101} = \alpha_{110} = 0$, so the pairs are good. Otherwise there are essentially three different possibilities, all bad: If $(i,j,m) = (1,2,4)$ then $\alpha_{101} = 0$, $\alpha_{100} = x_5x_9 \oplus x_6x_8$, $\alpha_{011} = x_9$; if $(i,j,m) = (1,2,6)$ then $\alpha_{010} = x_4x_9$, $\alpha_{011} = x_7$, $\alpha_{100} = x_5x_9$, $\alpha_{101} = x_8$; if $(i,j,m) = (1,5,9)$ then $\alpha_{111} = 1$, $\alpha_{110} = 0$, $\alpha_{010} = x_3x_7$.

70. (a) $x_1\wedge((x_5\wedge x_9)\oplus(x_6\wedge x_8)) \oplus x_2\wedge((x_6\wedge x_7)\oplus(x_4\wedge x_9)) \oplus x_3\wedge((x_4\wedge x_8)\oplus(x_5\wedge x_7))$.

(b) $x_1\wedge((x_5\wedge x_9)\vee(x_6\wedge x_8)) \vee x_2\wedge((x_6\wedge x_7)\vee(x_4\wedge x_9)) \vee x_3\wedge((x_4\wedge x_8)\vee(x_5\wedge x_7))$.

(c) Let $y_1 = x_1\wedge x_5\wedge x_9$, $y_2 = x_1\wedge x_6\wedge x_8$, $y_3 = x_2\wedge x_6\wedge x_7$, $y_4 = x_2\wedge x_4\wedge x_9$, $y_5 = x_3\wedge x_4\wedge x_8$, $y_6 = x_3\wedge x_5\wedge x_7$. The function $f(y_1,\ldots,y_6) = [y_1+y_2+y_3 > y_4+y_5+y_6]$ can be evaluated in 15 further steps with two full adders and a comparator; but there is a 14-step solution: Let $z_1 = (y_1\oplus y_2)\oplus y_3$, $z_2 = (y_1\oplus y_2)\vee(y_1\oplus y_3)$, $z_3 = (y_4\oplus y_5)\oplus y_6$, $z_4 = (y_4\oplus y_5)\vee(y_4\oplus y_6)$. Then $f = (z_1\oplus(z_2\wedge(\bar z_4\oplus(z_1\vee z_3))))\wedge(\bar z_3\vee z_4)$. Furthermore $y_1y_2y_3 = 111 \iff y_4y_5y_6 = 111$; so there are don't-cares, leading to an 11-step solution: $f = ((\bar z_1\wedge z_3)\vee\bar z_4)\wedge z_2$. The total cost is $12 + 11 = 23$.

(The author knows of no way by which a computer could discover such an efficient chain in a reasonable amount of time, given only the truth table of f. But perhaps an even better chain exists.)

71. (a) $P(p) = 1 - 12p^2 + 24p^3 + 12p^4 - 96p^5 + 144p^6 - 96p^7 + 24p^8$, which is $\frac{11}{32} + \frac{9}{2}\epsilon^2 - 3\epsilon^4 - 24\epsilon^6 + 24\epsilon^8$ when $p = \frac12 + \epsilon$.

(b) There are $N = 2^{n-3}$ sets of eight values $(f_0,\ldots,f_7)$, each of which yields good pairs with probability $P(p)$. So the answer is $1 - P(p)^N$.

(c) The probability is $\binom{N}{r}P(p)^r(1-P(p))^{N-r}$ that exactly r sets succeed; and in such a case t trials will find good pairs with probability $(r/N)^t$. The answer is therefore $1 - \sum_{r=0}^{N}\binom{N}{r}P(p)^r(1-P(p))^{N-r}(r/N)^t = 1 - P(p)^t + O(t^2/N)$.

(d) $\sum_{r=0}^{N}\binom{N}{r}P(p)^r(1-P(p))^{N-r}\sum_{j=0}^{t-1}(r/N)^j = (1-P(p)^t)/(1-P(p)) + O(t^3/N)$.

72. The probability in exercise 71(a) becomes $P(p) + (72p^3 - 264p^4 + 432p^5 - 336p^6 + 96p^7)r + (60p^2 - 240p^3 + 456p^4 - 432p^5 + 144p^6)r^2 + (-48p^2 + 144p^3 - 216p^4 + 96p^5)r^3 + (-36p^2 + 24p^3 + 12p^4)r^4 + (48p^2 - 24p^3)r^5 - 12p^2r^6$. If $p = q = (1-r)/2$, this is $(11 + 48r + 36r^2 - 144r^3 - 30r^4 + 336r^5 - 348r^6 + 144r^7 - 21r^8)/32$; for example, it's $7739/8192 \approx 0.94$ when $r = 1/2$.

73. Consider the Horn clauses $1\wedge 2\Rightarrow 3$, $1\wedge 3\Rightarrow 4$, ..., $1\wedge(n-1)\Rightarrow n$, $1\wedge n\Rightarrow 2$, and $i\wedge j\Rightarrow 1$ for $1<i<j\le n$. Suppose $|Z|>1$ in a decomposition, and let i be minimum such that $x_i\in Z$. Also let j be minimum such that $j>i$ and $x_j\in Z$. We cannot have $i>1$, since $i\wedge j\Rightarrow 1$ in that case. Thus $i=1$, and $x_j\in Z$ for $2\le j\le n$.

74. Suppose we know that no nontrivial decomposition exists with $x_1\in Z$ or $\cdots$ or $x_{i-1}\in Z$; initially $i=1$. We hope to rule out $x_i\in Z$ too, by choosing j and m cleverly. The Horn clauses $i\wedge j\Rightarrow m$ reduce to Krom clauses $j\Rightarrow m$ when i is asserted. So we essentially want to use Tarjan's depth-first search for strong components, in a digraph with arcs $j\Rightarrow m$ that may or may not exist.

When exploring from vertex j, first try $m=1$, ..., $m=i-1$; if any such implication $i\wedge j\Rightarrow m$ succeeds, we can eliminate j and all its predecessors from the digraph for i. Otherwise, test if $j\Rightarrow m$ for any such eliminated vertex m. Otherwise test unexplored vertices m. Otherwise try vertices m that have already been seen, favoring those near the root of the depth-first tree.

In the example $f(x)=(\det X)\bmod 2$, we would successively find $1\wedge 2\not\Rightarrow 3$, $1\wedge 2\Rightarrow 4$, $1\wedge 4\Rightarrow 3$, $1\wedge 3\Rightarrow 5$, $1\wedge 5\Rightarrow 6$, $1\wedge 6\Rightarrow 7$, $1\wedge 7\Rightarrow 8$, $1\wedge 8\Rightarrow 9$, $1\wedge 9\Rightarrow 2$ (now $i\leftarrow 2$); $2\wedge 3\not\Rightarrow 1$, $2\wedge 3\Rightarrow 4$, $2\wedge 4\not\Rightarrow 1$, $2\wedge 4\not\Rightarrow 5$, $2\wedge 4\Rightarrow 6$, $2\wedge 6\Rightarrow 1$ (now 3, 4, and 6 are eliminated from the digraph for 2), $2\wedge 5\Rightarrow 1$ (and 5 is eliminated), $2\wedge 7\not\Rightarrow 1$, $2\wedge 7\Rightarrow 3$ (7 is eliminated), $2\wedge 8\Rightarrow 1$, $2\wedge 9\Rightarrow 1$ (now $i\leftarrow 3$); $3\wedge 4\not\Rightarrow 1$, $3\wedge 4\Rightarrow 2$, $3\wedge 5\Rightarrow 1$, etc.

75. This function is 1 at only two points, which are complementary. So it is indecomposable; yet the pairs (58) are *never* bad when $n>3$. Every partition (Y,Z) will therefore be a candidate for decomposition.

Similarly, if f is decomposable with respect to (Y,Z), the indecomposable function $f(x)\oplus S_{0,n}(x)$ will act essentially like f in the tests. (A method to deal with *approximately decomposable functions* should probably be provided in a general-purpose decomposability tester.)

76. (a) Let $a_l=[i\ge l]$ for $0\le l\le 2^m$. The cost is $\le 2t(m)$, as observed in answer 38(b); and in fact, the cost can be reduced to $2^{m+1}-2m-2$ with $\Theta(m)$ depth. Furthermore the function $[i\le j]=(\bar{\imath}_1\wedge j_1)\vee((i_1\equiv j_1)\wedge[i_2\ldots i_m\le j_2\ldots j_m])$ can be evaluated with $4m-3$ gates. After computing $x\oplus y$, each z_l costs $2^{m+1}+1=O(n/\log n)$.

(b) Here the cost is at most $C(g_0)+\cdots+C(g_{2^m})\le(2^m+1)(2^{2^m}/(2^m+O(m)))$ by Theorem L, because each g_l is a function of 2^m inputs.

(c) If $i\le j$ we have $z_l=x$ for $l\le i$ and $z_l=y$ for $l>i$; hence $f_i(x)=c_0\oplus\cdots\oplus c_i$ and $f_j(y)=c_{j+1}\oplus\cdots\oplus c_{2^m}$. If $i>j$ we have $z_l=y$ for $l\le i$ and $z_l=x$ for $l>i$; hence $f_j(y)=c_0\oplus\cdots\oplus c_j$ and $f_i(x)=c_{i+1}\oplus\cdots\oplus c_{2^m}$.

(d) The functions $b_l=[j<l]$ can be computed for $0\le l\le 2^m$ in $O(2^m)$ steps, as in (a). So we can compute F from $(c_0,\ldots,c_{2^m})$ with $O(2^m)$ further gates. Step (b) therefore dominates the cost, for large m.

(e) $a_0=1$, $a_1=i$, $a_2=0$; $b_0=0$, $b_1=j$, $b_2=1$; $d=[i\le j]=\bar{\imath}\vee j$; $m_l=a_l\oplus d$, $z_{l0}=x_0\oplus(m_l\wedge(x_0\oplus y_0))$, $z_{l1}=x_1\oplus(m_l\wedge(x_1\oplus y_1))$, for $l=0,1,2$; $c_0=z_{01}$; $c_1=z_{10}\wedge\bar{z}_{11}$; $c_2=z_{20}\vee z_{21}$; $c'_l=c_l\wedge(d\equiv a_l)$, $c''_l=c_l\wedge(d\equiv b_l)$, for $l=0,1,2$; and finally $F=(c'_0\oplus c'_1\oplus c'_2)\vee(c''_0\oplus c''_1\oplus c''_2)$.

The net cost (29 after obvious simplifications) is, of course, outrageous in such a small example. But one wonders if a state-of-the-art automatic optimizer would be able to reduce this chain to just 5 gates.

[This result is a special case of more general theorems in *Matematicheskie Zametki* **15** (1974), 937–944; *London Math. Soc. Lecture Note Series* **169** (1992), 165–173.]

77. Given a shortest such chain for f_n or $\bar{f}_n$, let $U_l = \{i \mid l = j(i) \text{ or } l = k(i)\}$ be the "uses" of x_l, and let $u_l = |U_l|$. Let $t_i = 1$ if $x_i = x_{j(i)} \vee x_{k(i)}$, otherwise $t_i = 0$. We will show that there's a chain of length $\le r-4$ that computes either f_{n-1} or $\bar{f}_{n-1}$, by using the following idea: If variable x_m is set to 0 or 1, for any m, we can obtain a chain for f_{n-1} or $\bar{f}_{n-1}$ by deleting all steps of U_m and modifying other steps appropriately. Furthermore, if $x_i = x_{j(i)} \circ x_{k(i)}$ and if either $x_{j(i)}$ or $x_{k(i)}$ is known to equal t_i when x_m has been set to 0 or 1, then we can also delete the steps U_i. (Throughout this argument, the letter m will stand for an index in the range $1 \le m \le n$.)

Case 1: $u_m = 1$ for some m. This case cannot occur in a shortest chain. For if the only use of x_m is $x_i = \bar{x}_m$, eliminating this step would change $f_n \leftrightarrow \bar{f}_n$; and otherwise we could set the values of $x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n$ to make x_i independent of x_m, contradicting $x_{n+r} = f_n$ or $\bar{f}_n$. Thus every variable must be used at least twice.

Case 2: $x_l = \bar{x}_m$ for some l and m, where $u_m > 1$. Then $x_i = x_l \circ x_k$ for some i and k, and we can set $x_m \leftarrow \bar{t}_i$ to make x_i independent of x_k. Eliminating steps U_m, U_l, and U_i then removes at least 4 steps, except when $u_l = u_i = 1$ and $u_m = 2$ and $x_j = x_m \circ x_i$; but in that case we can also eliminate U_j.

Case 3: $u_m \ge 3$ for some m, and not Case 2. If $i, j, k \in U_m$ and $i < j < k$, set $x_m \leftarrow t_k$ and remove steps i, j, k, U_k.

Case 4: $u_1 = u_2 = \cdots = u_n = 2$, and not Case 2. We may assume that the first step is $x_1 = x_1 \circ x_2$, and that $x_l = x_1 \circ x_k$ for some $k < l$.

Case 4.1: $k > 0$. Then $k > 1$. If $u_k = 1$, set $x_1 \leftarrow t_l$ and remove steps 1, k, l, U_l. Otherwise set $x_2 \leftarrow t_1$; this forces $x_k = \bar{t}_l$, and we can remove steps 1, k, l, U_k.

Case 4.2: $x_l = x_1 \circ x_m$. Then we must have $m = 2$; for if $m > 2$ we could set $x_2 \leftarrow t_1$, $x_m \leftarrow t_l$, and make x_r independent of x_1. Hence we may assume that $x_1 = x_1 \wedge x_2$, $x_2 = x_1 \vee x_2$. Setting $x_1 \leftarrow 0$ allows us to remove U_0 and U_1; setting $x_1 \leftarrow 1$ allows us to remove U_0 and U_2. Thus we're done unless $u_1 = u_2 = 1$.

If $x_p = \bar{x}_1$, set $x_1 \leftarrow 0$ and remove 1, 2, p, U_p; if $x_q = \bar{x}_2$, set $x_1 \leftarrow 1$ and remove 1, 2, q, U_q. Otherwise $x_p = x_1 \circ x_u$ and $x_q = x_2 \circ x_v$, where x_u and x_v do not depend on x_1 or x_2. But that's impossible; it would allow us to set $x_3, \ldots, x_n$ to make $x_u = t_p$, then $x_2 \leftarrow 1$ to make x_r independent of x_1.

[*Problemy Kibernetiki* **23** (1970), 83–101; **28** (1974), 4. With similar proofs, Red'kin showed that the shortest AND-OR-NOT chains for the functions '$x_1 \ldots x_n < y_1 \ldots y_n$' and '$x_1 \ldots x_n = y_1 \ldots y_n$' have lengths $5n-3$ and $5n-1$, respectively.]

78. [*SICOMP* **6** (1977), 427–430.] Say that y_k is *active* if $k \in S$. We may assume that the chain is normal and that $\|S\| > 1$; the proof is like Red'kin's in answer 77:

Case 1: Some active y_k is used more than once. Setting $y_k \leftarrow 0$ saves at least two steps and yields a chain for a function with $\|S\| - 1$ active values.

Case 2: Some active y_k appears only in an AND gate. Setting $y_k \leftarrow 0$ eliminates at least two steps, unless this AND is the final step. But it can't be the final step, because $y_k = 0$ makes the result independent of every other active y_j.

Case 3: Like Case 2 but with an OR or NOT-BUT or BUT-NOT gate. Setting $y_k \leftarrow c$ for some appropriate constant c has the desired effect.

Case 4: Like Case 2 but with XOR. The gate can't be final, since the result should be independent of y_k when $(x_1 \ldots x_m)_2$ addresses a different active value y_j. So we can eliminate two steps by setting y_k to the function defined by the *other* input to XOR.

79. (a) Suppose the cost is $r < 2n-2$; then $n > 1$. If each variable is used exactly once, two leaves must be mates. Therefore some variable is used at least twice. Pruning it away produces a chain of cost $\le r-2$ on $n-1$ variables, having no mates.

(Incidentally, the cost is at least $2n-1$ if every variable is used at least twice, because at least $2n$ uses of variables must be connected together in the chain.)

(b) Notice that $S_{0,n} = \bigwedge_{u - v}(u \equiv v)$ whenever the edges u — v form a free tree on $\{x_1, \ldots, x_n\}$. So there are many ways to achieve cost $2n-3$.

Any chain of cost $r < 2n-3$ must have $n > 2$ and must contain mates u and v. By renaming and possibly complementing intermediate results, we can assume that $u = 1$, $v = 2$, and that $f(x_1, \ldots, x_n) = g(x_1 \circ h(x_3, \ldots, x_n), x_2, \ldots, x_n)$, where $\circ$ is $\wedge$ or $\oplus$.

Case 1: $\circ$ is AND. We must have $h(0, \ldots, 0) = h(1, \ldots, 1) = 1$, for otherwise $f(x_1, x_2, y, \ldots, y)$ wouldn't depend on x_1. Therefore $f(x_1, \ldots, x_n) = h(x_3, \ldots, x_n) \wedge g(x_1, x_2, \ldots, x_n)$ can be computed by a chain of the same cost in which 1 and 2 are mates and in which the path between them has gotten shorter.

Case 2: $\circ$ is XOR. Then $f = f_0 \vee f_1$, where $f_0(x_1, \ldots, x_n) = (x_1 \equiv h(x_3, \ldots, x_n)) \wedge g(0, x_2, \ldots, x_n)$ and $f_1(x_1, \ldots, x_n) = (x_1 \oplus h(x_3, \ldots, x_n)) \wedge g(1, x_2, \ldots, x_n)$. But $f = S_{0,n}$ has only two prime implicants; so there are only four possibilities:

Case 2a: $f_0 = f$. Then we can replace $x_1 \oplus h$ by 0, to get a chain of cost $\le r-2$ for the function $g(0, x_2, \ldots, x_n) = S_{0,n-1}(x_2, \ldots, x_n)$.

Case 2b: $f_1 = f$. Similar to Case 2a.

Case 2c: $f_0(x) = x_1 \wedge \cdots \wedge x_n$ and $f_1(x) = \bar{x}_1 \wedge \cdots \wedge \bar{x}_n$. In this case we must have $g(0, x_2, \ldots, x_n) = x_2 \wedge \cdots \wedge x_n$ and $g(1, x_2, \ldots, x_n) = \bar{x}_2 \wedge \cdots \wedge \bar{x}_n$. Replacing h by 1 therefore yields a chain that computes f in $< r$ steps.

Case 2d: $f_0(x) = \bar{x}_1 \wedge \cdots \wedge \bar{x}_n$ and $f_1(x) = x_1 \wedge \cdots \wedge x_n$. Similar to Case 2c.

Applying these reductions repeatedly will lead to a contradiction. Similarly, one can show that $C(S_0 S_n) = 2n-2$. [*Theoretical Computer Science* **1** (1976), 289–295.]

80. [*Mathematical Systems Theory* **10** (1977), 323–336.] Without loss of generality, $a_0 = 0$ and the chain is normal. Define U_l and u_l as in answer 77. We may assume by symmetry that $u_1 = \max(u_1, \ldots, u_n)$.

We must have $u_1 \ge 2$. For if $u_1 = 1$, we could assume further that $x_{n+1} = x_1 \circ x_2$; hence two of the three functions $S_\alpha(0, 0, x_3, \ldots, x_n) = S_{\alpha''}$, $S_\alpha(0, 1, x_3, \ldots, x_n) = S_{'\alpha'}$, $S_\alpha(1, 1, x_3, \ldots, x_n) = S_{''\alpha}$ would be equal. But then S_α would be a parity function, or $S_{'\alpha'}$ would be constant.

Therefore setting $x_1 = 0$ allows us to eliminate the gates of U_1, giving a chain for $S_{\alpha'}$ with at least 2 fewer gates. It follows that $C(S_\alpha) \ge C(S_{\alpha'}) + 2$. Similarly, setting $x_1 = 1$ proves that $C(S_\alpha) \ge C(S_{'\alpha}) + 2$.

Three cases arise when we explore the situation further:

Case 1: $u_1 \ge 3$. Setting $x_1 = 0$ proves that $C(S_\alpha) \ge C(S_{\alpha'}) + 3$.

Case 2: $U_1 = \{i, j\}$ and operator $\circ_j$ is canalizing (namely, AND, BUT-NOT, NOT-BUT, or OR). Setting x_1 to an appropriate constant forces the value of x_j and allows us to eliminate $U_1 \cup U_j$; notice that $i \notin U_j$ in an optimum chain. So either $C(S_\alpha) \ge C(S_{\alpha'}) + 3$ or $C(S_\alpha) \ge C(S_{'\alpha}) + 3$.

Case 3: $U_1 = \{i, j\}$ and $\circ_i = \circ_j = \oplus$. We may assume that $x_i = x_1 \oplus x_2$ and $x_j = x_1 \oplus x_k$. If $u_j = 1$ and $x_l = x_j \oplus x_p$, we can restructure the chain by letting $x_j = x_k \oplus x_p$, $x_l = x_1 \oplus x_j$; therefore we can assume that either $u_j \ne 1$ or $x_l = x_j \circ x_p$ for some canalizing operator $\circ$. If $U_2 = \{i, j'\}$, we can assume similarly that $x_{j'} = x_2 \oplus x_{k'}$ and that either $u_{j'} = 1$ or $x_{l'} = x_{j'} \circ' x_{p'}$ for some canalizing operator $\circ'$. Furthermore we can assume by symmetry that x_j does not depend on $x_{j'}$.

If x_k does not depend on x_i, let $f(x_3, \ldots, x_n) = x_k$; otherwise let $f(x_3, \ldots, x_n)$ be the value of x_k when $x_i = 1$. By setting $x_1 = f(x_3, \ldots, x_n)$ and $x_2 = \bar{f}(x_3, \ldots, x_n)$, or vice versa, we make x_i and x_j constant, and we obtain a chain for the nonconstant

function $S_{'\alpha'}$. We can, in fact, ensure that x_l is constant in the case $u_j = 1$. We claim that at least five gates of this chain (including x_i and x_j) can be eliminated; hence $C(S_\alpha) \geq C(S_{'\alpha'}) + 5$. The claim is clearly true if $|U_i \cup U_j| \geq 3$.

We must have $|U_i \cup U_j| > 1$. Otherwise we'd have $p = i$, and x_k would not depend on x_i, so S_α would be independent of x_1 with our choice of x_2. Therefore $|U_i \cup U_j| = 2$.

Case 3a: $U_j = \{l\}$. Then x_l is constant; we can eliminate x_i, x_j, and $U_i \cup U_j \cup U_l$. If the latter set contains only two elements, then $x_q = x_i \circ x_l$ is also constant and we eliminate U_q. Since $S_{'\alpha'}$ isn't constant, we won't eliminate the output gate.

Case 3b: $U_i \subseteq U_j$, $|U_j| = 2$. Then $x_q = x_i \circ x_j$ for some q; we can eliminate x_i, x_j, and $U_j \cup U_q$. The claim has been proved.

(b) By induction, $C(S_k) \geq 2n + \min(k, n-k) - 3 - [n = 2k]$, for $0 < k < n$; $C(S_{\geq k}) \geq 2n + \min(k, n+1-k) - 4$, for $1 < k < n$. The easy cases are $C(S_0) = C(S_n) = C(S_{\geq 1}) = C(S_{\geq n}) = n - 1$; $C(S_{\geq 0}) = 0$. (According to Figs. 9 and 10, these bounds are optimum for $k = \lceil n/2 \rceil$ when $n \leq 5$. All known results are consistent with the conjecture that $C(S_k) = C(S_{\geq k})$ for $k \geq n/2$.)

81. If some variable is used more than once, we can set it to a constant, decreasing n by 1 and decreasing c by ≥ 2. Otherwise the first operation must involve x_1, because $y_1 = x_1$ is the only output that doesn't need computation; making x_1 constant decreases n by 1, c by ≥ 1, and d by ≥ 1. [*J. Algorithms* **7** (1986), 185–201.]

82. (62) is false.

(63) reads, "For all numbers m there's a number n such that $m < n + 1$"; it is true because we can take $m = n$.

(64) fails when $n = 0$ or $n = 1$, because the numbers in these formulas are required to be nonnegative integers.

(65) says that, if b exceeds a by 2 or more, there's a number ab between them. Of course it's true, because we can let $ab = a + 1$.

(66) was explained in the text, and it too is true. Notice that '$\wedge$' takes precedence over '$\vee$' and '$\equiv$' takes precedence over '$\Leftrightarrow$', just as '+' takes precedence over '$\geq$' and '<' over '$\wedge$' in (65); these conventions reduce the need for parentheses in sentences of L.

(67) says that, if A contains at least one element n, it must contain a minimum element m (an element that's less than or equal to all of its elements). True.

(68) is similar, but m is now a maximum element. Again true, because all sets are assumed to be finite.

(69) asks for a set P with the property that $[0 \in P] = [3 \notin P]$, $[1 \in P] = [4 \notin P]$, ..., $[999 \in P] = [1002 \notin P]$, $[1000 \in P] \neq [1003 \notin P]$, $[1001 \in P] \neq [1004 \notin P]$, etc. It's true if (and only if) $P = \{x \mid x \bmod 6 \in \{1, 2, 3\} \text{ and } 0 \leq x < 1000\}$.

Finally, the subformula $\forall n\,(n \in C \Leftrightarrow n + 1 \in C)$ in (70) is another way of saying that $C = \emptyset$, because C is finite. Hence the parenthesized formula after $\forall A\,\forall B$ is a tricky way to say that $A = \emptyset$ and $B \neq \emptyset$. (Stockmeyer and Meyer used this trick to abbreviate statements in L that involve long subformulas more than once.) Statement (70) is true because an empty set doesn't equal a nonempty set.

83. We can assume that the chain is normal. Let the canalizing steps be $y_1, \ldots, y_p$. Then $y_k = \alpha_k \circ \beta_k$ and $f = \alpha_{p+1}$, where α_k and β_k are $\oplus$'s of some subsets of $\{x_1, \ldots, x_n, y_1, \ldots, y_{k-1}\}$; at most $n+k-2$ $\oplus$'s are needed to compute them, combining common terms first. Hence $C(f) \leq p + \sum_{k=1}^{p+1}(n + k - 2) = (p+1)(n + p/2) - 1$.

84. Argue as in the previous answer, with $\vee$ or $\wedge$ in place of $\oplus$. [N. Alon and R. B. Boppana, *Combinatorica* **7** (1987), 15–16.]

85. (a) A simple computer program shows that 13744 are legitimate and 19024 aren't. (An illegitimate family of this kind has at least 8 members; one such is {00, 0f, 33, 55, ff, 15, 3f, 77}. Indeed, if the functions $x_1 \vee x_2$ (3f), $x_2 \vee x_3$ (77), and $(x_1 \vee x_2) \wedge x_3$ (15) are present in a legitimate family L, then $x_2 \sqcup 15 = 33 \mid 15 = 37$ must also be in L.)

(b) The projection and constant functions are obviously present. Define $A^* = \bigcap\{B \mid B \supseteq A \text{ and } B \in \mathcal{A}\}$, or $A^* = \infty$ if no such set B exists. Then we have $\lceil A \rceil \sqcap \lceil B \rceil = \lceil A \cap B \rceil$ and $\lceil A \rceil \sqcup \lceil B \rceil = \lceil (A \cup B)^* \rceil$.

(c) Abbreviate the formulas as $\hat{x}_l \subseteq x_l \vee \bigvee_{i=n+1}^{l} \delta_i$, $x_l \subseteq \hat{x}_l \vee \bigvee_{i=n+1}^{l} \epsilon_i$, and argue by induction: If step l is an AND step, $\hat{x}_l = \hat{x}_j \sqcap \hat{x}_k \subseteq \hat{x}_j \wedge \hat{x}_k \subseteq (x_j \vee \bigvee_{i=n+1}^{l} \delta_i) \wedge (x_k \vee \bigvee_{i=n+1}^{l} \delta_i) = x_l \vee \bigvee_{i=n+1}^{l} \delta_i$; $x_l = x_j \wedge x_k \subseteq (\hat{x}_j \vee \bigvee_{i=n+1}^{l-1} \epsilon_i) \wedge (\hat{x}_k \vee \bigvee_{i=n+1}^{l-1} \epsilon_i) = (\hat{x}_j \wedge \hat{x}_k) \vee \bigvee_{i=n+1}^{l-1} \epsilon_i$, and $\hat{x}_j \wedge \hat{x}_k = \hat{x}_l \vee \epsilon_l$. Argue similarly if step l is an OR step.

86. (a) If S is an r-family contained in the $(r+1)$-family S', clearly $\Delta(S) \subseteq \Delta(S')$.

(b) By the pigeonhole principle, $\Delta(S)$ contains elements u and v of each part, whenever S is an r-family. And if $\Delta(S) = \{u, v\}$, we certainly have $u \mathrel{—} v$.

(c) The result is obvious when $r = 1$. There are at most $r - 1$ edges containing any given vertex u, by the "strong" property. And if $u \mathrel{—} v$, the edges *disjoint* from $\{u, v\}$ are strongly $(r-1)$-closed; so there are at most $(r-2)^2$ of them, by induction. Thus there are at most $1 + 2(r-2) + (r-2)^2$ edges altogether.

(d) Yes, by exercise 85(b), if $r > 1$, because strongly r-closed graphs are closed under intersection. All graphs with ≤ 1 edges are strongly r-closed when $r > 1$, because they have no r-families containing distinct edges.

(e) There are $\binom{n}{3}$ triangles $x_{ij} \wedge x_{ik} \wedge x_{jk}$, only $n-2$ of which are contained in any term x_{uv} of $\hat{f}$. Hence the minterms for at most $(r-1)^2(n-2)$ triangles are contained in $\hat{f}$, and the others must be contained in one of the functions $\epsilon_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \wedge x_{k(i)})$. Such a term has the form $T = (\lceil G \rceil \sqcap \lceil H \rceil) \oplus (\lceil G \rceil \wedge \lceil H \rceil) = (\lceil G \rceil \wedge \lceil H \rceil) \wedge \overline{\lceil G \cap H \rceil}$, where G and H are strongly r-closed; we will prove that T contains at most $2(r-1)^3$ triangles.

A triangle $x_{ij} \wedge x_{ik} \wedge x_{jk}$ in T must involve some variable (say x_{ij}) of $\lceil G \rceil$ and some variable (say x_{ik}) of $\lceil H \rceil$, but no variable of $\lceil G \cap H \rceil$. There are at most $(r-1)^2$ choices for ij; and then there are at most $2(r-1)$ choices for k, since H has at most $r-1$ edges touching i and at most $r-1$ edges touching j.

(f) There are 2^{n-1} complete bigraphs obtained by coloring 1 red, coloring other vertices either red or blue, and letting $u \mathrel{—} v$ if and only if u and v have opposite colors. By the first formula in exercise 85(c), the minterms B for every such graph must be contained in one of the terms $T = \delta_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \vee x_{k(i)}) = \lceil (G \cup H)^* \rceil \wedge \overline{\lceil G \cup H \rceil}$. (For example, if $n = 4$ and vertices $(2, 3, 4)$ are (red, blue, blue), then $B = \bar{x}_{12} \wedge x_{13} \wedge x_{14} \wedge x_{23} \wedge x_{24} \wedge \bar{x}_{34}$.) A minterm B is contained in T if and only if, in the coloring for B, some edge of $(G \cup H)^*$ has vertices of opposite colors, but all edges of $G \cup H$ are monochromatic. We will prove that T includes at most $2^{n-r} r^2$ such B.

Let G be any graph, and $T = \lceil G^* \rceil \wedge \overline{\lceil G \rceil}$. The following (inefficient) algorithm can be used to find G^*: If there's an r-family S with $|\Delta(S)| < 2$, stop with $G^* = \infty$. Otherwise, if $\Delta(S) = \{u, v\}$ and $u \mathrel{—\!\!\!\!/} v$, add the edge $u \mathrel{—} v$ to G and repeat.

At most 2^{n-r} bipartite minterms B have monochromatic $\{u_j, v_j\}$ for $1 \le j \le r$ when $|\Delta(S)| < 2$. And when $\Delta(S) = \{u, v\}$ there are 2^{n-r-1} with monochromatic $\{u_j, v_j\}$ and bichromatic $\{u, v\}$. So we want to show that the algorithm for G takes fewer then $2r^2$ iterations when G is strongly r-closed.

For $k \ge 1$, let $u_k \mathrel{—} v_k$ be the first new edge added to G that is disjoint from $\{u_j, v_j\}$ for $1 \le j < k$. At most r such edges exist, by "strongness"; and each of them

is followed by at most $2r-3$ new edges that touch u_j or v_j. So the total number of steps to find G^* is at most $r(2r-2)+1 < 2r^2$.

(g) Exercise 84 tells us that $q < \binom{p}{2} + (p+1)\binom{n}{2}$. Thus we have either $2(r-1)^3 p \ge \binom{n}{3} - (r-1)^2(n-2)$ or $\binom{p}{2} + (p+1)\binom{n}{2} > 2^{r-1}/r^2$. Both lower bounds for p are

$$\frac{1}{12}\Bigl(\frac{n}{6\lg n}\Bigr)^3\Bigl(1+O\Bigl(\frac{\log\log n}{\log n}\Bigr)\Bigr) \qquad \text{when} \qquad r=\Bigl\lceil \lg\Bigl(\frac{n^6}{186624(\lg n)^4}\Bigr)\Bigr\rceil.$$

[Noga Alon and Ravi B. Boppana, *Combinatorica* **7** (1987), 1–22, proceeded in this way to prove, among other things, the lower bound $\Omega(n/\log n)^s$ for the number of $\wedge$'s in any monotone chain that decides whether or not G has a clique of fixed size $s \ge 3$.]

87. The entries of X^3 are at most n^2 when X is a 0–1 matrix. A Boolean chain with $O(n^{\lg 7}(\log n)^2)$ gates can implement Strassen's matrix multiplication algorithm 4.6.4–(36), on integers modulo $2^{\lfloor \lg n^2 \rfloor + 1}$.

88. There are 1,422,564 such functions, in 716 classes with respect to permutation of variables. Algorithm L and the other methods of this section extend readily to ternary operations, and we obtain the following results for optimum median-only computation:

$C(f)$	Classes	Functions	$C_m(f)$	Classes	Functions	$L(f)$	Classes	Functions	$D(f)$	Classes	Functions
0	1	7	0	1	7	0	1	7	0	1	7
1	1	35	1	1	35	1	1	35	1	1	35
2	2	350	2	2	350	2	2	350	2	13	5670
3	9	3885	3	9	3885	3	8	3745	3	700	1416822
4	48	42483	4	48	42483	4	38	35203	4	1	30
5	201	406945	5	188	391384	5	139	270830	5	0	0
6	353	798686	6	253	622909	6	313	699377	6	0	0
7	99	169891	7	69	134337	7	176	367542	7	0	0
8	2	282	8	2	2520	8	34	43135	8	0	0
9	0	0	9	0	0	9	3	2310	9	0	0
10	0	0	10	0	0	10	0	0	10	0	0
11	0	0	∞	143	224654	11	1	30	11	0	0

S. Amarel, G. E. Cooke, and R. O. Winder [*IEEE Trans.* **EC-13** (1964), 4–13, Fig. 5b] conjectured that the 9-operation formula

$$\langle x_1x_2x_3x_4x_5x_6x_7\rangle = \langle x_1\langle\langle x_2x_3x_5\rangle\langle x_2x_4x_6\rangle\langle x_3x_4x_7\rangle\rangle\langle\langle x_2x_5x_6\rangle\langle x_3x_5x_7\rangle\langle x_4x_6x_7\rangle\rangle\rangle$$

is the best way to compute medians-of-7 via medians-of-3. But the "magic" formula

$$\langle x_1\langle x_2\langle x_3x_4x_5\rangle\langle x_3x_6x_7\rangle\rangle\langle x_4\langle x_2x_6x_7\rangle\langle x_3x_5\langle x_5x_6x_7\rangle\rangle\rangle\rangle$$

needs only 8 operations; and in fact the shortest chain needs just seven steps:

$$\langle x_1x_2x_3x_4x_5x_6x_7\rangle = \langle x_1\langle x_2\langle x_5x_6x_7\rangle\langle x_3\langle x_5x_6x_7\rangle x_4\rangle\rangle\langle x_5\langle x_2x_3x_4\rangle\langle x_6\langle x_2x_3x_4\rangle x_7\rangle\rangle\rangle.$$

The interesting function $f(x_1,\ldots,x_7) = (x_1\wedge x_2\wedge x_4) \vee (x_2\wedge x_3\wedge x_5) \vee (x_3\wedge x_4\wedge x_6) \vee (x_4\wedge x_5\wedge x_7) \vee (x_5\wedge x_6\wedge x_1) \vee (x_6\wedge x_7\wedge x_2) \vee (x_7\wedge x_1\wedge x_3)$, whose prime implicants correspond to the projective plane with 7 points, is the toughest of all: Its minimum length $L(f)=11$ and minimum depth $D(f)=4$ are achieved by the remarkable formula

$$\langle\langle x_1x_4\langle x_4x_5x_6\rangle\rangle\langle x_3x_6\langle x_1\langle x_2x_3x_7\rangle\langle x_2x_5x_6\rangle\rangle\rangle\langle x_2x_7\langle x_1\langle x_5x_2x_4\rangle\langle x_5x_3x_7\rangle\rangle\rangle\rangle.$$

And the following even more astonishing chain computes it optimally:

$$x_8=\langle x_1x_2x_3\rangle,\quad x_9=\langle x_1x_4x_6\rangle,\quad x_{10}=\langle x_1x_5x_8\rangle,\quad x_{11}=\langle x_2x_7x_8\rangle,$$
$$x_{12}=\langle x_3x_9x_{10}\rangle,\quad x_{13}=\langle x_4x_5x_{12}\rangle,\quad x_{14}=\langle x_6x_{11}x_{12}\rangle,\quad x_{15}=\langle x_7x_{13}x_{14}\rangle.$$

INDEX AND GLOSSARY

When an index entry refers to a page containing a relevant exercise, see also the *answer* to that exercise for further information. An answer page is not indexed here unless it refers to a topic not included in the statement of the exercise.

⌣, *see* smile.
0–1 matrices, 20, 32–35, 44, 46, 125, 130, 179, 200, *see also* Adjacency matrices.
0–1 principle, 68.
0-preserving functions, *see* Normal Boolean functions.
2-coloring, 17, 22–23, 41.
2-monotonic functions, *see* Regular Boolean functions.
2-partite graphs, *see* Bipartite graphs.
2-variable functions, 47–50, 79–80.
  table, 49.
2^m-way multiplexer, 109, 127.
2CNF, 57, 72, 86–87, 91, *see also* Krom functions.
2SAT, 57, 60–62, 72, 86.
2SAT functions, *see* 2CNF.
3-coloring, 39, 42, 149.
3-variable functions, 63, 104–105, 156.
  table, 78.
3CNF, 56.
3SAT, 56, 162.
4-coloring, 17, 39, 149.
4-cycles, 69.
4-variable functions, 98–105, 112–114, 122, 126, 129.
5-variable functions, 105–106, 126, 191.
$\alpha(H)$ (independence number of a graph or hypergraph), 35, 44.
γ (Euler's constant), as source of "random" data, 136.
νx (sideways sum), 77, 90, 108, 126, 158, 185, 187.
π (circle ratio), as source of "random" data, 76, 98, 118, 128, 136, *see also* Pi function.
$\Sigma(f)$ (true-vector sum), 76–77, 92, 95.
ϕ (golden ratio), 134.
  as source of "random" data, 136.
$\chi(H)$ (chromatic number of a graph or hypergraph), 35, 39, 44, 46.
$\omega(G)$ (clique number of a graph), 35, 44.

a-codes, 82, *see* Asterisk codes for subcubes.
Abelian (commutative) groups, 152.
Absorption laws, 50.
Acyclic: Containing no cycles, 15, 31-32.
Acyclic digraphs, 169.
Addition, binary, 107–108, 127–128.
Addition modulo 3 and 5, 129.
Addition modulo 4, 126.
Addition table modulo n, 36.
Adjacency lists, 21–22.
Adjacency matrices, 19–20, 26, 27, 40–41, 43, 123, 133, 148, 153.
Adjacent vertices, 13.
Affine Boolean functions, 95–96, 182–183.
Affirmation (⊤), 49.
Ahrens, Wilhelm Ernst Martin Georg, 136.
Aiken, Howard Hathaway, 104.
Ajtai, Miklós, 91.
Akers, Sheldon Buckingham, Jr., 87.
Alekseyev, Valery Borisovich (Алексеев, Валерий Борисович), 179.
All-0s matrix, 27, 145.
All-1s matrix, 26, 27, 145, 146.
All-equal function ($S_{0,n}$), 131.
Alon, Noga (נגה אלון), 175, 198, 200.
Alphabetic order, 38.
Amarel, Saul, 200.
Amir, Yair (יאיר עמיר), 174.
Anagrams, 139.
Analysis of algorithms, 41, 82–84, 130.
AND (∧), 48–51, 53, 57, 63, 81.
  bitwise (&), 58, 74, 82, 84, 156, 158.
AND operation, bitwise, 22.
AND gates (∧), 32, 33, 97.
  with vacuum tubes, 104.
AND-OR chains, 125, 132–133.
AND-OR–NOT chains, 131, *see also* Canalizing chains.
Andersen, Lars Døvling, 136.
Antisymmetric digraphs, 62, 91.
Appel, Kenneth Ira, 17.
Apportionment, 8.
Approximately decomposable functions, 195.
Arabic mathematics, 135–136.
Arc lists, 21–22.
Arc variables, 21, 23.
Arcs in a graph, 18–23.
  as edges, 18.
ARCS(v) (first arc of a vertex), 21.
Arithmetic progressions, 38.
Aschbacher, Michael George, 143.
Ashenhurst, Robert Lovett, 117, 120.
Aspvall, Bengt Ingemar, 87, 164.
Associative block designs, 56.
Associative laws, 27, 28, 42, 50, 65, 68, 80, 156, 159, 185, 186, 189.
Asterisk codes for subcubes, 82, 84.
Asterisks, 54, 82–84, 155.
Asymptotic methods, 109–112, 128–129, 157, 160, 169, 174, 177–178.
Automated deduction, 167.

Automorphisms, 14–15, 39, 45, 143, 147, 152–153.
Availability polynomials, 80, 81, 84, 93, 155.
Avann, Sherwin Parker, 89.

b-codes, 82, *see* Bit codes for subcubes.
Babbage, Charles, 116, 188, 193, 194.
Babbage, Henry Provost, 188.
Bach, Johann Sebastian, xii.
Backward-computation principle, 102.
Bad pairs, 118–120, 130–131.
Ball, Michael Owen, 161.
Bandelt, Hans-Jürgen, 170.
Bang, Thøger Sophus Vilhelm, 134.
Barbará Millá, Daniel, 88.
Barbour, Andrew David, 152.
Barnard, Robert, 10.
Baron, Gerd, 135.
Barycentric coordinates, 25, 88.
Bassanio of Venice, 1.
Baugh, Charles Richmond, 172, 173, 178.
Bears, California Golden, 31.
Bell Telephone Laboratories, 115.
Bent functions, 96.
Berge, Claude, 34.
Berlekamp, Elwyn Ralph, 192.
Bernays, Paul Issak, 53.
Bernstein, Benjamin Abram, 154.
Betweenness, 65, 89–90.
bi_book graphs, 23.
bi_lisa graphs, 24.
Biggs, Norman Linstead, 15.
Bigraphs, 17, *see* Bipartite graphs.
Bijunctive clauses, *see* Krom clauses.
Binary addition, 107–108, 127–128.
Binary comparison function, 120, 194–196.
Binary decoder, 109, 186.
Binary majorization lattices, 92–93.
Binary number system, 47, 75–76, 80, 82, 90, 92, 175.
Binary operator: A function of two variables, 49.
Binary recurrences, 108, 109, 126, 168, 186.
Binary strings, 54, 67, 90, 92–93.
Binary trees, 85, 97, 98, 190.
  complete, 81.
Binary-coded decimal digits, 114.
Bioch, Jan Corstiaan, 166.
Bipartite graphs, 17, 22–25, 35, 39, 41, 43, 120, 133, 142, 149, 194.
  corresponding to hypergraphs, 33, 44.
Bipartite hypergraphs, 149.
Bipartite matching, 125.
Bipartiteness testing, 22–23.
Birkhoff, Garrett, 167.
Bishop moves on a chessboard, 25, 26, 31.
Bit codes for subcubes, 82, 84–85.
Bitburger Brauerei, x.
Bitwise operations, 47, 74, 82, 100–102, 126, 154, 182, 199.
  AND (&), 22, 58, 74, 82, 84, 156, 158.
  medians, 67, 71, 72, 91, 167, 168.
  OR (|), 50, 74, 84.
  saturating subtraction ($\dot{-}$), 84.
  XOR (⊕), 90, 158.
Blake, Archie, 159.
Block designs, 179.
Blum, Norbert Karl, 122.
board graphs, 25, 26, 31, 41.
Bocheński, Józef (= Innocenty) Maria, 49.
Bollobás, Béla, 179.
Bondy, John Adrian, 14, 153.
book graphs, 23.
Boole, George, 48, 52, 189.
Boolean binary operators, 47–51, 80, 87, 92.
  table, 49.
Boolean chains, 96–133.
  AND-OR, 125, 132–133.
  AND-OR-NOT, 126, 131, 190.
  canalizing, 126, 132, 190.
  definition of, 96.
  monotone, 125, 132–133.
  of 3 variables, 104–105.
  of 4 variables, 98–105, 112–114, 122, 126, 129.
  of 5 variables, 105–106, 126, 191.
  of many variables, 109–112, 117–133.
  optimization of, 121–122, 195.
  with several outputs, 107–109, 112–117, 121–122, 126–130.
Boolean functions, 33, 47–95.
  affine, 95–96, 182–183.
  bent, 96.
  canalizing, 78, 79, 95.
  duals of, 154, 157, 172.
  enumeration of, 79.
  Horn, 58, 79, 95.
  Krom, 60, 72, 79, 81, 95.
  majority, 63, 68, 76, 169, *see* Medians.
  monotone, 156, *see* Monotone Boolean functions.
  of 2 variables, 47–50, 79–80.
  of 3 variables, 63, 78, 156.
  random, 56, 83, 160.
  regular, 93, 178.
  self-dual, *see* Self-dual Boolean functions.
  symmetric, 77–79, 94–95.
  threshold, 75–77, 79, 92–95, 186.
  unate, 156.
Boolean games, 86.
Boolean values, 32.
Bootstrapping, 185, 190.
Boppana, Ravi Babu, 198, 200.
Boros, Endre, 120, 161.
Bose, Raj Chandra (রাজ চন্দ্র বসু), 5.
Bossen, Douglas Craig, 139.
Bottom-up synthesis, 103–105, 126, 183.

Boyer, Robert Stephen, 46.
Brayton, Robert King, 122.
Breadth-first search, 70.
Brent, Richard Peirce, 188.
Brette, Jean, 135.
Brewster, George, 9.
Brightwell, Graham Richard, 179.
Brinkmann, Gunnar, 149.
Brooks, Rowland Leonard, 145.
Brown, John Wesley, 136.
Brown, William Gordon, 143.
Brualdi, Richard Anthony, 137.
Bruck, Richard Hubert, 138.
Büchi, Julius Richard, 124.
Buddies, 82.
Bui, Alain, 2.
Burley (= Burleigh), Walter, 51.
BUT-NOT gates ($\overline{\supset}$), 97, 100, 110.

C language, 22, 50.
$C(f)$, 97, 111–112, *see* Cost of a Boolean function.
$C(f_1 \ldots f_m)$, 107.
$C^+(f)$, 132, 133.
$C_m(f)$, 102–103, 126, 200.
C_n (cycle of order n), 13, 28, 39, 41.
$C_{\vec{n}}$ (oriented cycle of order n), 18, 41.
California Golden Bears, 31.
Canalizing chains, 126, 132, 190.
Canalizing functions, 78, 79, 95.
Canalizing operators, 123, 126, 183, 197.
Cancellation laws, 76, 80.
Canonical forms, *see* Full conjunction normal form, Full disjunctive normal form, Integer multilinear representation, Multilinear representation of a Boolean function.
Cardinal, Stanford, 31.
Cares, 129, 194, *see also* Don't-cares.
Carroll, Lewis (= Dodgson, Charles Lutwidge), 10–12, 48, 79, 140.
Carry bits, 107, 127, 128, 189.
Cartesian product of graphs, 27–28, 42–44, 67, 146.
Cat's game, 86, 115, 117, 193.
Categorical product of graphs, 28, *see* Direct product of graphs.
Cayley, Arthur, digraphs, 45.
  graphs, 45, 152.
Censorship, 10–11.
Chambers, Ephraim, v.
Chandra, Ashok Kumar (अशोक कुमार चन्द्रा), 159, 176.
Characteristic polynomial of a Boolean function, 155, *see* Integer multilinear representation.
Chase, Philip John, sequence, 94.
Chaucer, Geoffrey, 139.
Chess, 163.
Chessboard-like graphs,
  bishop moves, 25, 26, 31.
  generalized piece moves, 41.
  king moves, 43.
  knight moves, 15, 25.
  queen moves, 26, 44.
  rook moves, 26, 41.
Chien, Robert Tien Wen (錢天聞), 139.
Christie Mallowan, Agatha Mary Clarissa Miller, 18–19.
Chromatic index, *see* Edge-chromatic number.
Chromatic number $\chi(G)$, 35, 39, 44, 46.
Chow, Chaw Kong (周紹康), 76.
  parameters $N(f)$ and $\Sigma(f)$, 76–77, 92, 95.
Christmas tree patterns, 177.
Chung Graham, Fan Rong King (鍾金芳蓉), 168.
Chvátal, Václav, 14.
  graph, 14, 39, 44, 149.
CI-nets, 72–74, 91.
Circuits, Boolean, 97, *see* Boolean chains.
Clapham, Christopher Robert Jasper, 148.
Clause: A disjunction of literals, 54, 81.
Clausen, Thomas, 5.
Claw graph, 141.
Clique covers, 35.
Clique number $\omega(G)$, 35, 44.
Cliques, 35, 44, 200.
Closure under intersections, 57, 132.
Closure under medians, 72.
Clustering, vii.
CMath: Concrete Mathematics, a book by R. L. Graham, D. E. Knuth, and O. Patashnik.
CNF, 53, *see* Conjunctive normal form.
Coalitions, 169.
Codewords, b-ary, 38.
Coding theory, 139.
Cographs, 42.
Cohen, Philip Michael, 140.
Coins, biased, 96.
Colleges, 31.
Coloring of graphs, 17, 35, 42, 44, 46, 120–121.
Coloring of hypergraphs, 32, 35, 44.
Combinational complexity, 97, 111–112, *see* Cost of a Boolean function.
Combinatorial explosion, v.
Combinatorics, 1–7, *see also* Graphs.
Comedy festival, 60–62, 86.
Commutative laws, 28, 50, 65, 90, 159, 185.
Comparator modules, 72–74, 91, 127, 170.
Comparator-inverter networks, 72–74, 91.
Comparison function, binary, 92, 120, 194–196.
Compiler technology, 58.

Complement, of a Boolean function, 49, 55, 57, 78–79, 99–100, 107, 154, 184.
  of a graph, 26, 27, 35, 41, 42, 146.
  of a simple digraph, 42.
  of an r-uniform hypergraph, 32.
Complementation laws, 50–51.
Complete bigraphs ($K_{m,n}$), 17, 26, 39, 42, 142.
Complete binary trees, 81.
Complete binary tries, 38–39.
Complete bipartite graphs, 17, 26, 39, 42, 133, 142, 199.
Complete digraphs (J_n), 18, 142, 144.
Complete graphs (K_n), 13, 26–27, 39, 41–43, 145.
Complete k-partite graphs, 17, 26–27, 40, 44.
Complete r-uniform hypergraphs, 32.
  bipartite, 44.
Complete ternary tries, 140.
Complete tripartite graphs ($K_{m,n,r}$), 17, 42.
Completion of a matrix, 46.
Components, 16, 18, 26, 40, 42, 43.
Composition of functions, 189.
Composition of graphs, 28, *see* Lexicographic product of graphs.
Compositions of an integer, 25.
Condensation principle, 89.
Conditional-sum adders, 127–128.
Conjugate of a partition, 29–30, 43, 148.
Conjunction ($\wedge$), 49, *see* AND.
Conjunction of graphs, 28, *see* Direct product of graphs.
Conjunctive normal form (CNF), 53, 56–57, 72, 81, 85, 87, 97, 183, 191.
  full, 53.
  monotone, 81.
Conjunctive prime form, 54, 81.
Connected digraphs, 18.
Connected graphs, 16, 33, 43, 44.
Connectivity of a graph, viii, 150.
Consecutive 1s, 84, 86, 127, 128.
Consecutive arcs, 19.
Consensus, 83.
Consonants, 38.
Context-free grammar, 85.
Contiguous United States of America, 15, 34, 39–40.
Contradiction ($\perp$), 49.
Contrapositive, 61.
Control grids, 104.
Converse implication ($\subset$), 49.
Converse nonimplication ($\overline{\subset}$), 49, 80.
Converse of a digraph, 145.
Convex hull of points, 24, 68–69.
Convex sets, 68–69, 90.
Conway, John Horton, 192.
Cook, Stephen Arthur, 162.
Cooke, George Erskine, 200.
Coppersmith, Don, 152.
Core of a Horn function, 58, 86, 162, 164.
Cost of a Boolean function, 97, 107, 111–112, 126–132.
  statistics, 101, 105, 183–184.
Coteries, 88, 93.
Cover, Thomas Merrill, 13.
Covering in a lattice, 93.
Covering problems, 11.
  exact, 2, 7, 8, 35, 37, 135.
  minimum, 34–35, 44.
Crama, Yves Jean-Marie Mathieu Franz, 161.
Cretté de Palluel, François, 8.
Crossings in a diagram, 14, 150.
Cube graphs (k-cubes), 28, 41.
Cubes, 66, *see also* Hypercubes, Subcubes.
Cubic graphs, 14, 39, 151.
Curtis, Herbert Allen, 96, 120–121.
Cutler, Robert Brian, 158.
Cycle graph C_n, 13, 28, 39, 41.
Cycles, 13, 28, 39, 41, 42, 44.
  in a hypergraph, 33.
  of a permutation, 40, 42.
  oriented, 18, 19, 32, 40, 41.
Cylinders, 28, 41.

$D(f)$, 99, *see* Depth of a Boolean function.
$d^+(v)$ (out-degree of v), 18.
$d^-(v)$ (in-degree of v), 18.
$d(u,v)$ (distance in a graph), 16, 43.
  directed, 19.
  generalized, 16–17.
da Vinci, Leonardo, 9, 24.
Dags (directed acyclic graphs), 31–32.
Dahlheimer, Thorsten, 165.
Dancing links method, 2, 7, 8, 11, 135.
Data replication, 88.
Davies, Roy Osborne, 2, 134.
De Micheli, Giovanni, 122.
De Morgan, Augustus, 51.
  laws, 51, 81, 154.
de Palluel, François Cretté, 8.
de Polignac, Camille Armand Jules Marie, 15.
Decoder, binary, 109, 186.
Decomposition of functions, 117–121, 130–131.
Decomposition of partial functions, 120–121.
Decomposition or development laws, 51, 52.
Definite Horn clauses, 58–60, 86.
Definite Horn functions, 58, 85, 95.
Degree of a vertex, 14, 19, 39, 43, 44, 148.
Degree sequences, 29–31, 43, 46, 152–153.
Delaunay, Boris Nikolaevich (Делоне, Борис Николаевич), triangulation, 24.
Depth of a Boolean function, 99, 100, 124–128, 132.
  statistics, 101, 105, 183–184.
Depth-first search, 23, 41, 195.

Determinants, 40, 121, 130, 146, 152.
Diagonal matrices, 151, 152.
Diagonalization, 124.
Diagrams for digraphs, 18–19, 42.
Diagrams for graphs, 14–15, 26–28, 39, 42.
Diameter of a free median graph, 169.
Diameter of a graph, 16, 24, 39, 41, 42, 44, 145.
Dictionaries of English, v, 10, 34, 38, 47, 48, 140.
Digitized image, 24.
Digraphs, 18, *see* Directed graphs.
Dillon, John Francis, 180.
Direct product of graphs, 28, 42–43.
Direct product of matrices, 43.
Direct sum of graphs, 26–27, 42, 43.
Direct sum of matrices, 27, 43.
Directed acyclic graphs, 31–32.
Directed distance $d(u,v)$, 19.
Directed graphs, 12, 18–22, 40, 42, 146.
  complete, 18, 142, 144.
  components of, 18.
  converse of, 145.
  random, 25.
  representation of, 19–22.
  strong components of, 40, 61–62, 86, 142, 164, 195.
Directed hypergraphs, 44.
Directed join of digraphs, 26–27.
Discrete Fourier transforms, 94, 155.
Disjoint decomposition, 117–120.
Disjoint graphs, 26.
Disjoint sets, 25.
Disjunctive normal form (DNF), 53–55, 81, 85, 97, 191.
  full, 53–54, 81, 84, 176.
  irredundant, 95, 158, 160.
  monotone, 81, 82.
  orthogonal, 84–85, 92.
  shortest, 55, 82, 83, 95.
Disjunctive prime form, 54–55, 64, 71.
Distance $d(u,v)$ in a graph, 16, 43.
  generalized, 16–17.
Distance of a code, 38.
Distinct columns, 33.
Distinct rows, 46.
Distributed systems, 88.
Distributive lattices, 92, 170, 173.
Distributive laws, 43, 48, 50, 80, 87, 93, 125, 146, 156, 157.
  for medians, 65, 67, 87, 90, 167.
Divide and conquer, 109, 185–188.
DNF, 53, *see* Disjunctive normal form.
Dodecahedron, 15.
Dodgson, Charles Lutwidge (= Lewis Carroll), 10–12, 48, 79, 140.
Dominance order, *see* Majorization lattices.
Dominated coteries, 88.
Don't-cares, 46, 114, 116, 120–122, 129, 131, 194.
Dot minus ($\dot{-}$), 49, 84.
Dot product of vectors, 12, 34, 37.
Doublets game, 11.
Doubly linked lists, 70.
Doutté, Edmond, 135.
Dowling, William Francis, 60.
Doyle, Arthur Conan, 1.
Dual identities, 154.
Dual of a Boolean function: $F^D(x) = \bar{F}(\bar{x})$, 154, 157, 172; *see also* Self-dual Boolean functions.
  computing monotone CNF from DNF, 157.
Dual of a hypergraph, 33, 35, 44.
Dudeney, Henry Ernest, 141.
Dunham, Bradford, 94, 176.
Dynamic programming, 154.

e, as source of "random" data, 136.
Early neighbors, 69, 90.
econ graphs, 31.
Edge-chromatic number $\chi(L(G))$, 44.
Edges in a graph, 11, 13.
  as arcs, 19, 21–22.
Edges in a hypergraph, 32–35.
Egiazarian, Karen, 178.
Eigenvalues of a matrix, 40.
Eigenvectors of a matrix, 40.
Ekin, Oya, 161.
Electoral districts, 8.
Electrical engineers, 96, 97, 105, 107, 109, 114, 122.
Elementary symmetric functions, 155.
Elgot, Calvin Creston, 124, 171.
Empty graphs ($\overline{K_n}$), 26, 27, 41–43, 46, 150.
English language, v, 9–10.
Enumeration of Boolean functions, 79.
  asymptotic, 177.
Equidistant cities, 44.
Equivalence operator ($\equiv$), 49–50, 154, 172.
Equivalence relations, 45–46, 138.
Equivalence under permutations, 78–79.
  and complementations, 78–79, 178.
Erdős, Pál (= Paul), 147, 148, 150, 169, 174.
Error-correcting codes, 37–38.
Euclidean distance, 10, 12.
Euclidean plane, 17.
Euler, Leonhard (Ейлеръ, Леонардъ = Эйлер, Леонард), 3–7, 36, 136.
Evaluation of Boolean functions, 96–133, *see* Boolean chains.
Even permutations, 40.
Evolution of random graphs, 25.
Ewing, Ann Catherine, 159.
Exact cover problems, 2, 7, 8, 35, 37, 135.
Exclusive disjunction ($\oplus$), 49, *see* XOR.
Existential quantifiers, 85, 87.
Expander graphs, 24.

Exponential growth: $2^{\Theta(n)}$.
Exponential time, 157.
Extended real numbers: Real numbers together with $-\infty$ and $+\infty$, 63.

Factorization of a graph, 28.
Failing units, 80.
Falsehood ($\perp$), 49, 63, 79.
Families of sets, 32, *see* Hypergraphs.
Families of subsets, 87–88.
Fan-in: The number of inputs to a gate, 97, 104, 124.
Fan-out: The number of uses of a gate, 97, 104, 189.
Feder, Tomás, 73, 170.
Fenwick, Peter McAulay, 188.
Fibonacci, Leonardo, of Pisa (= Leonardo filio Bonacii Pisano),
 numbers, 186, 188.
 strings, 36.
 threshold functions, 92, 125.
Fiduccia, Charles Michael, 152.
Fields, finite, 50, 138.
Final vertex of an arc, 18.
Finck, Hans-Joachim, 150.
Finikov, Boris Ivanovich (Фиников, Борис Иванович), 125.
Finite fields, 50, 138.
Finite state transducers, 128.
First-order predicate calculus, 164.
Fischer, Michael John, 127, 128.
Fischler, Martin Alvin, 175.
Fišer, Petr, 55.
Five-letter words, 9–12, 16, 38–39, 43.
Five-variable functions, 105–106, 126, 191.
Flows of money, 31.
Floyd, Robert W, iv.
 Lemma, vi.
Folland, Gerald Budge, iii.
Football scores, 31.
Footprints, 100–101, 113, 114, 122, 125, 126, 191–192.
Forcade, Rodney Warring, 152.
Forcing functions, 78.
Formula complexity, *see* Length of a Boolean function.
Four Color Theorem, 17.
Four-variable functions, 98–105, 112–114, 122, 126, 129.
Fourier, Jean Baptiste Joseph, transform, discrete, 94, 155.
Fraer, Ranan (רענן פרייר), 87.
Fredman, Michael Lawrence, 157.
Free median algebras, 70–71, 91.
Free systems, 155.
Free trees, 17, 44, 67, 91, 197.
Frequency of usage in English, 10.
Fridshal, Richard, 94, 176.
Fulkerson, Delbert Ray, 148.
Full adders, 107–108, 126, 184, 189, 194.
Full conjunctive normal form, 53.
Full disjunctive normal form, 53–54, 81, 84, 177.
Functional decomposition, 117–121, 130–131.
Funk, Isaac Kauffman, 48.

Gaddum, Jerry William, 150.
Gadzhiev, Makhach Mamaevich (Гаджиев, Махач Мамаевич), 177.
Galen, Claudius (Κλαύδιος Γαληνός), 49.
Gallai, Tibor, 147.
Gallier, Jean Henri, 60.
Games, 86, *see also* Tic-tac-toe.
games graphs, 31.
Ganter, Bernhard, 136.
García-Molina, Héctor, 88.
Gardner, Martin, 9, 11, 166, 193.
Gates, networks of, 32, 97, 104.
Gauß (= Gauss), Johann Friderich Carl (= Carl Friedrich), 5, 17.
Gégalkine (= Zhegalkin), Jean Jean (Жегалкин, Иван Иванович), 51, 155.
Generalized consensus, 83, 120.
Generalized toruses, 45–46.
Generator routines, 23–26, 30–32, 41.
Geometric nets, 37–39.
Gherardini, Lisa, *see Mona Lisa*.
Giant component of a graph, 16, 25, 39, 142.
Girth of a graph, 15, 24, 39–41, 44.
Glagolev, Valery Vladimirovich (Глаголев, Валерий Владимирович), 177.
Globally optimum solutions, 34–35.
Godfrey, Michael John, 36.
Golden Bears, California, 31.
Golomb, Solomon Wolf, 139.
Good pairs, 118, 130–131.
Gordon, Leonard Joseph, 139.
GOST cipher, 129.
Goto, Eiichi (後藤英一), 171, 173.
Græco-Latin squares, 4–5, 8, 36, 137.
Graham, Ronald Lewis (葛立恒), 90, 168, 203.
Grammar, context-free, 58.
Grant, Jeffrey Lloydd Jagton, 140.
Graph homomorphisms, 73.
Graph-paper graphs, 28.
Graph theory, introduction to, 13–19.
Graphical degree sequences, 29–31, 43, 46, 152–153.
Graphs, 11–35, 39–45.
 algebra of, 26–28, 42–45.
 bipartite, *see* Bipartite graphs.
 complete, 13, 26–27, 39, 41–43, 145.
 empty (null), 26, 27, 41–43, 46, 150.
 generators for, 23–26, 30–32, 41.
 labeled versus unlabeled, 15, 16, 152.
 of orders 3 and 4, 42, 46.
 products of, 27–28, 42–44, 146.
 random, 25, 41, 46.

regarded as digraphs, 19–22.
regular, 14, 24–25, 33, 40–44.
representation of, 19–22.
Gray, Frank, binary code, 135, 186.
Greatest lower bounds, 93.
Greedy algorithm, 113.
Greedy-footprint heuristic, 113–114, 122, 191–192.
Greek logic, 48–49.
Grid graphs, 28, 41.
triangular, 25, 145.
Gries, David Joseph, 153.
Gropes, 155.
Groth, Edward John, Jr., 8.
Grünbaum, Branko, 149.
Gualterus Burleus (= Walter Burley), 51.
Guilielmus ab Occam (= William of Ockham), 51.
gunion (union of SGB graphs), 26.
Gurvich, Vladimir Alexander (Гурвич, Владимир Александрович), 120, 157.
Guthrie, Francis, 17.
Guy, Richard Kenneth, 192.

Hachtel, Gary Deane, 122.
Hadamard, Jacques Salomon
inequality, 171.
matrix, 179.
transform, 155.
Hagauer, Johann (= Hans), 69.
Haken, Armin, 162.
Haken, Wolfgang, 17.
Half adders, 107–108.
Hall, Marshall, Jr., ix, 134, 138.
Hamilton, William Rowan, 15.
cycles and paths, 15, 85.
Hamiltonian graphs, 15, 44, 141.
Hammer, Péter László (= Peter Leslie = Ivănescu, Petru Ladislav), 120, 161.
Hamming, Richard Wesley, distance, 12, 28, 37–38, 90, 168.
Harary, Frank, x, 18, 142.
Hardware versus software, 97, 188.
Harvard University Computation Laboratory, 126, 184.
Håstad, Johan Torkel, 94.
Havel, Václav (mathematician), 29.
Havel, Václav (playwright and statesman), 8–9.
Hebraic-Græco-Latin squares, 36, 137.
Hebrew letters, 36.
Hedayat, Samad (= Abdossamad, عبدالصمد هدايت), 136, 138.
Heinen, Franz, 17.
Hell, Pavol, 74.
Hellerman, Leo, 104–105.
Hensel, Kurt Wilhelm Sebastian, 146.
Hexadecimal notation for truth tables, 105, 132, 191.
Hight, Stuart Lee, 120.
Highways, 31.
Hindman, Neil Bruce, 169.
Hlavička, Jan, 55.
Hoffman, Alan Jerome, 143.
Holmes, Thomas Sherlock Scott, 1.
Holton, Derek Alan, 151.
Homer (Ὅμηρος), 9.
Homomorphisms in median algebras, 67.
Homomorphisms of graphs, 74.
Horn, Alfred, 57.
clauses, 57, 85, 86, 119, 167, 195.
functions, 58, 79, 95.
functions, renamed, 87.
satisfiability, 60, 85–87.
Horton, Robert Elmer, 140.
Hotels and comedians, 60–62, 86.
Hsiao, Ben Mu-Yue (蕭慕岳 = 萧慕岳), 139.
Huffman, David Albert, 94.
Hugo, Victor Marie, 23.
Huntington, Edward Vermilye, 167.
Hurwitz, Adolf, 146.
Hyperarcs, 149.
Hypercubes, 54, 74, *see* n-cube.
retracts of, 74, 91.
subgraphs of, 90.
Hyperedges, 32.
Hyperforests, 44.
Hypergraphs, 32–35, 44.
Hyperrectangles ($P_{n_1} \square \cdots \square P_{n_m}$), 28, 67.
Hypotheses, 59–60.

I (identity matrix), 26, 146.
Ibaraki, Toshihide (茨木俊秀), 120, 161, 165, 166.
IBM Type 650 computer, v.
Ibn al-Ḥājj, Muḥammad ibn Muḥammad (محمد بن محمد بن الحاج), 135.
id of an SGB graph, 12, 22.
ID(g), 22.
Ideals in a median algebra, 65.
Identity matrix, 26.
IEEE Transactions, ix.
If-then-else function (mux), 96, 181, 185, 187, 189.
Igarashi, Yoshihide (五十嵐善英), 176.
Image, digitized, 24.
Implicants, 53–54, 81, *see also* Prime implicants.
Implication (⊃), 48–49.
Implicit data structure, 21–22.
Imrich, Wilfried, 28, 69, 168.
In-degree of a vertex, 18, 19, 41, 43.
Incidence matrix of a graph or hypergraph, 33, 35, 44.
Inclusion-exclusion principle, 159.
Inclusive disjunction (∨), 49, *see* OR.
Incremental changes, 90.
Incompatible columns, 120.

Independence number $\alpha(H)$ of a graph or hypergraph, 35, 44.
Independent vertices, 34, 35, 44.
Induced subgraphs, 13, 18, 39, 42, 43, 46.
Induced subhypergraphs, 32.
Infinity, point at, 24.
Initial vertex of an arc, 18.
Integer multilinear representation, 80, 84, 94, 155, 174.
Integer programming, vii.
Internet, ii, iv, viii, ix, 10, 140.
ITE, *see* If-then-else function.
Interpolating polynomials, 156.
Intersecting families of sets, 88, 174.
Intervals in graphs, 66, 67, 89.
Intervals in median algebras, 65–66.
Intruders, 36.
Inverse permutation, 20.
Inverter gates, 32, *see* Complement of a Boolean function.
Inverter modules, 72–74, 91, 170.
Irredundant DNFs, 94–95, 158, 160.
Islamic mathematics, 135–136.
Isolated vertices: Vertices of degree 0, 25, 142, 145.
Isometric: Distance-preserving, 90.
  subgraphs, 90, 91, 170.
Isomorphic graphs, 13–15, 28, 39, 152–153.
  directed, 18.
Isotone functions, 156.
Istrate, Gabriel, 163.
Ivănescu, Petru Ladislav (= Hammer, Peter Leslie), 120, 161.

J (all-ones matrix), 26, 27, 145, 146.
J_n (complete digraph of order n), 18, 142, 144.
Jaillet, Christophe André Georges, 2.
Janson, Carl Svante, 46, 152.
Jevons, William Stanley, 48, 51, 155.
Jha, Pranava Kumar (प्रणव कुमार झा), 69.
Johnson, Samuel, v.
Join of graphs, 26–27, 150.
Just, Winfried, 179.
Juxtaposition of graphs, 26, *see* Direct sum of graphs.

k-ary trees, 124.
k-colorable graphs or hypergraphs, 17, 32, 35, 42, 44.
k-edge-colorable graphs, 26, 42, 44.
k-in-a-row function, 127.
k-partite graphs or hypergraphs, 17, 32, 35, 42, 44.
  complete, 17, 26–27, 40, 44.
K_n (complete graph of order n), 13, 26–27, 41–43, 145.
$K_n^{(r)}$ (complete r-uniform hypergraph), 32.
$K_n^{\rightarrow}$ (transitive tournament of order n), 18, 27, 40, 41.
$K_{3,3}$ (utilities graph), 17, 39, 42, 138.
$K_{m,n}$ (complete bipartite graph), 17, 26, 39, 42, 142.
$K_{m,n}^{(r)}$ (complete r-uniform bipartite hypergraph), 44.
$K_{n_1,\ldots,n_k}$ (complete k-partite graph), 17, 26, 40, 44.
Kameda, Tiko (= Tsunehiko) (亀田恒彦), 165, 174.
Karoński, Michał, 152.
Karp, Richard Manning, 121.
Karpiński (= Karpinski), Marek Mieczysław, 165.
Kauffman, Stuart Alan, 78.
Kautz, William Hall, 175.
Ke Zhao (= Chao Ko, 柯召), 174.
Keister, William, 115.
Kelly, Paul Joseph, 148.
Kempe, Alfred Bray, 141.
Kernel of a graph, *see* Maximal independent sets.
Khachiyan, Leonid Genrikhovich (Хачиян, Леонид Генрихович), 157.
Khrapchenko, Valerii Mikhailovich (Храпченко, Валерий Михайлович), 188.
King moves on a chessboard, 43.
Kingwise torus, 44.
Kirkman, Thomas Penyngton, 15.
Kiss, Stephen Anthony, 167.
Klavžar, Sandi, 28, 69.
Klein, Peter, 127.
Kleine Büning, Hans Gerhard, 163–165.
Kleitman, Daniel J (Isaiah Solomon), 148, 169.
Kline, John Robert, 167.
Knapsacks, 75.
Knight moves on a chessboard, 15, 25.
Knuth, Donald Ervin (高德纳), i, ii, iv, ix, xii, 7, 9–10, 12, 32, 62, 105, 115, 136, 157, 168, 188, 203.
Knuth, John Martin, *see* Truth.
Ko, Chao (= Ke Zhao, 柯召), 174.
Koch, John Allen, 17.
Kogan, Alexander (Коган, Александр Юрьевич), 120, 161.
Kolibiar, Milan, 167.
Komlós, János, 91.
Kőnig, Dénes, 17.
Konvalina, John, 179.
Korshunov, Aleksey Dmitrievich (Коршунов, Алексей Дмитриевич), 177.
Kowalewski, Arnold, 145.
Krajecki, Michaël, 2.
Kratochvíl, Jan, 46, 152.

Krichevsky, Rafail Evseevich (Кричевский, Рафаил Евсеевич), 190.
Krom, Melven Robert, 62, 164.
  clauses, 57, 72, 85, 87, 195.
  functions, 60, 72, 79, 81, 95, *see also* 2CNF.
  satisfiability, 57, 60–62, 72, 86.
Kronecker, Leopold, product, 146.
Kung, Hsiang Tsung (孔祥重), 188.

$L(f)$, 99, *see* Length of a Boolean function.
$L(G)$ (line graph of G), 26, 42.
Labeled graphs, 15, 152.
Labels of graph vertices, 67–74, 90.
Laborde, Jean-Marie, 159.
LADDERS program, 32.
Ladner, Richard Emil, 127, 128.
Lähdesmäki, Harri, 178.
Lamport, Leslie B., 88.
Landau, Hyman Garshin, 142.
Langford, Charles Dudley, 7, 9, 135.
  pairs, 1–3, 8, 36.
  triples, 36.
Las Vegas hotels, 60–62, 86.
Latch gates, 32.
Late neighbors, 69.
Latin squares, 3–8, 36–38, 136, 137, 149.
Lattices, *see also* Majorization lattices.
  distributive, 92, 170, 173.
Leader, Imre, 179.
Leahy, Francis Theodore, Jr. (= Ted), 134.
Least upper bounds, 93.
Left complementation (⊏̄), 49, 80.
Left projection (L), 49, 63, 80, 154.
Legitimate lattices of functions, 132, 133.
Lehmer, Derrick Henry, 39.
Leighton, Robert Eric, 139.
Length of a Boolean function, 99, 103, 125, 192.
  statistics, 101, 105, 183–184.
Lenin, Vladimir Ilyich (Ленин, Владимир Ильич), 86.
Leonardo da Vinci, 9, 24.
Lettmann, Theodor August, 163, 164.
Lexicographic order, 3, 38, 41, 75, 85, 86, 165.
Lexicographic product of graphs, 28, 42–43, 146.
Liang, Franklin Mark, 105.
Lillywaite, Peregrine, 10.
Lindström, Bernt Lennart Daniel, 173.
Line graph of a graph, 26, 35, 42, 142, 146, 148.
Linear inequalities, 171–172.
Linear ordering, 161.
Linear polynomials, 52.
Linear programming, vii, 92.
Linear subgraphs, 142.
Linear time, 57, 82.
Linked allocation, 21.
Literals, 53.
Lloyd, Edward Keith, 15.
Lo Shu magic square, 192.
LOC (memory location), 22.
Local optimizations, 121.
Locally optimal solutions, 34–35.
Logic, 48–51, 123–124, 132.
Long distributive law, 65, 67, 89.
Loops from a vertex to itself, 13, 18, 19, 41, 148.
Lower bounds on combinational complexity, 103–104, 109–112, 122–124, 131–132.
Loyd, Walter (= "Sam Loyd, Jr."), 1.
Lupanov, Oleg Borisovich (Лупанов, Олег Борисович), 110, 112, 129, 185, 190, 192.

$m\times n$ cylinders, 28, 41.
$m\times n$ grids, 28, 41.
$m\times n$ rook graphs, 26, 41.
$m\times n$ toruses, 28, 41.
  directed, 41.
M(g) (the number of arcs in an SGB graph), 22.
$M(n)$ (binary majorization lattice), 173.
MacNeish, Harris Franklin, 5.
Magic Fifteen, 129.
Magic squares, 36, 192.
Majority element, 46.
Majority functions, 63, 68, 104, 139, *see* Medians.
Majority law, 65, 89.
Majority of odd, *see* Median of odd.
Majorization lattices, of binary vectors, 92–94.
  of n-tuples, 30, 148, 173.
Makino, Kazuhisa (牧野和久), 174.
Mann, Henry Berthold, 136.
Marcisová, Tamara, 167.
Matching in a graph, 35, 44, 119.
Markowsky, George, 159, 176.
MATE (the converse arc), 21–22.
Mates in a Boolean chain, 131.
Mates of arcs, 21–22.
Mathon, Rudolf Anton, 136.
Matrix multiplication, 20, 146, 200.
max (maximum operator), 63–64.
Maximal cliques, 44.
Maximal independent sets, 34, 44.
Maximal intersecting families, 88.
Maximal planar graphs, 39.
Maximal subcubes, 54–55, 82–83.
Maximal versus maximum, 34–35.
Maximum independent sets, 34–35.
Maximum matchings, 44.
Maxterms, *see* Minclauses.
Mayr, Ernst Wilhelm, 170.
McCluskey, Edward Joseph, Jr., 55.
McCulloch, Warren Sturgis, 75.

McCune, William Walker, Jr., 167.
McKay, Brendan Damien, 137.
McKellar, Archie Charles, 118, 120, 131.
McManus, Christopher DeCormis, 38.
Median algebras, 64–67, 89.
Median chains, 133.
Median expansion formula, 87.
Median graphs, 67–74, 90, 169.
Median labels, 67–74, 168.
Median of odd, 64, 75–76, 91–92, 94.
  of five, 64, 68, 71, 76, 77, 87, 91.
Median of seven, 133.
Median operation, 102, 125, 133.
Median sets, 72–74, 91.
Medians, 62–74, 87–91.
  bitwise, 67, 71, 72, 91, 167, 168.
Mems: Memory accesses, 2, 6.
Menon, Vairelil Vishwanath (വൈരേലിൽ വിശ്വനാഥ മേനോൻ), 145.
Meringer, Markus Reinhard, 149.
METAFONT, 12.
Metropolis, Nicholas Constantine (Μητρόπολης, Νικόλαος Κωνσταντίνου), 176.
Meyer, Albert Ronald da Silva, viii, 123, 124, 198.
Meyerowitz, Aaron David, 89, 169.
Meynert, Alison, 137.
Mezei, Jorge Esteban (= György István), 92.
Miiller, Henry Sedwick, 64.
miles graphs, 31, 44.
Mileto, Franco, 160.
Miller, Donald John, 147.
Mills, Burton E., 156, 159.
Milnor, John Willard, 88, 166.
min (minimum operator), 63–64.
Minclauses, 53.
Minimal versus minimum, 34–35.
Minimal vertex covers, 157.
Minimum-memory evaluation, 101–103, 106, 125, 126, 186.
Minimum vertex covers, 34–35, 44.
Minnick, Robert Charles, 77, 172.
Minterms, 52–54, 109, 111, 126, 155.
MIP-years, 2.
Mirror pairs, 38.
Misra, Jayadev (ଜୟଦେବ ମିଶ୍ର), 153.
Mixed-radix majorization lattices, 173.
Mixed-radix numbers, 174.
MMIX computer, ii, iv, viii, 41.
MMIXAL assembly language, 41.
Mnemonics, 146.
Mod 2 canonical form, *see* Multilinear representation of a Boolean function.
Modular arithmetic, 135.
Mona Lisa, 9, 24, 31.
Monadic logic: Logic with only unary operators, 123–124, 132.
Monotone Boolean chains, 125, 132–133.
Monotone Boolean functions, 55, 63, 79, 81, 82, 85, 87, 95, 156.
  computing CNF from DNF, 157.
  self-dual, 63–64, 70, 79, 87–89, 92–93, 133.
  shellability of, 84–85, 157, 161, 171.
  threshold, 75–76, 92.
Monotone complexity, 106, 125, 132–133.
Moon, John Wesley, 150.
Moore, Eliakim Hastings, 138.
Moore, J Strother, 46.
Moore, Ronald Williams, 157.
Morgenstern, Oskar, 169.
Morreale, Eugenio, 158.
Morris, Scot Anderson, 38.
Moser, Leo, 150.
Mulder, Henry Martyn, 167.
Muller, David Eugene, 187.
Multilevel logic synthesis, 122.
Multigraphs, 13, 19–21, 40, 41, 44, 146, 152.
Multihypergraphs, 148.
Multilinear representation of a Boolean function, 52, 80, 96, 130, 156, 183; *see also* Integer multilinear representation.
Multilinked data structures, 60.
Multipairs, 19.
Multiple outputs, 107–109, 112–117, 121–122, 126–130.
Multiplexer, 2-way, *see* Mux.
  2^m-way, 109, 127, 131.
Multiplication, 8.
Multisets, 12, 18, 19.
Muroga, Saburo (室賀三郎), 77, 158, 171–173, 178.
Murphy's Law, 148.
Museum of Science and Industry, 115.
Musical graph, 44, 45.
Mutual exclusion, 88.
Mutually orthogonal latin squares, 37–38.
Mux (multiplex) operation, 96, 181, 185, 187, 189.
Myrvold, Wendy Joanne, 137.

n-ary Boolean functions, 51–55.
n-ary strings, 37.
n-cube: The 2^n points $(x_1, \ldots, x_n)$ with $x_j = 0$ or $x_j = 1$ in each coordinate position, 28, 41, 73, 129.
  subcubes of, 54, 82–84, 155, 160, 161, 176.
N(g) (the number of vertices in an SGB graph), 22, 143.
Name servers, 88.
NAME(v) (the name of a vertex), 21.
NAND ($\bar\wedge$), 49–51, 80–81, 183.
  with vacuum tubes, 104.
Nanocomputer simulation, 32.
Nebeský, Ladislav, 167.
Nechiporuk, Eduard Ivanovich (Нечипорук, Эдуард Иванович), 192.
Neighboring vertices, 13.
Nemhauser, George Lann, 161.

Nešetřil, Jaroslav, 146.
Networks: Graphs or digraphs together with auxiliary data, 31–32, 44.
Neumann, John von (= Margittai Neumann János), 169.
Neural networks, 75.
NEXT(a) (the next arc with the same initial vertex), 21.
Nigmatullin, Roshal' Gabdulkhaevich (Нигматуллин, Рошаль Габдулхаевич), 177.
Nodes in SGB format, 21–23, 143.
Nonconjunction ($\overline{\wedge}$), 49, *see* NAND.
Nondisjunction ($\overline{\vee}$), 49, *see* NOR.
Nonimplication ($\overline{\supset}$), 49.
NOR gates ($\overline{\vee}$), 49–51, 104–105.
Nordhaus, Edward Alfred, 150.
Normal Boolean functions, 100, 102, 110, 113, 125, 184, 196.
Normal forms, *see* Full conjunction normal form, Full disjunctive normal form, Integer multilinear representation, Multilinear representation of a Boolean function.
Normalization, 100, 181.
NOT gates, 32, 33.
NOT-BUT gates ($\overline{\subset}$), 97, 100.
Notation, 26, 132, 146.
Notational conventions, iv.
  for Boolean binary operators, 48–50.
  for symmetric Boolean functions, 77.
  $\overline{G}$ (complementation), 26, 32, 42.
  $G \cong G'$ (isomorphism), 14.
  $G \mid V'$ (induced subgraph), 13.
  $G \setminus e$ (edge removal), 13.
  $G \setminus v$ (vertex removal), 13.
  $[u\,..\,v]$ (closed interval), 65.
  $x \subseteq y$ (componentwise $\leq$), 55.
  $\langle xyz \rangle$ (median), 62–63.
  $\langle x_1 \ldots x_{2k-1} \rangle$ (median), 64.
  $x?\ y\colon z$, 96, *see* Mux (multiplex) operation.
Noughts and crosses, *see* Tic-tac-toe.
Novels, 9, 23.
NP-complete problems, viii, 35, 55, 157.
Null graphs ($\overline{K_n}$), 26, 27, 41–43, 46, 150.

O (all-zeros matrix), 27.
Oblivious sorting, 72.
Ockham, William of (= Guilielmus ab Occam), 51.
Odd permutations, 40.
Odd product of graphs, 28, 42–43, 146.
ODNFs, 84–85, 92.
Ofman, Yuri Petrovich (Офман, Юрий Петрович), 189.
Optimal versus optimum, 34–35.
Optimization of Boolean chains, 121–122, 195.
Optimum Boolean evaluation, 103–105.
Optimum coteries, 93.
OR (inclusive or, $\vee$), 48–51, 53, 63, 81.
  bitwise ($|$), 50, 74, 84.
OR gates ($\vee$), 32, 33, 97.
  with vacuum tubes, 104.
Order of a digraph, 18.
Order of a graph, 13, 44.
Order of a latin square, 37.
Order of an orthogonal array, 37.
Ore, Øystein, ix, 42.
Organ-pipe order, 135.
Oriented binary trees, 190.
Oriented cycles, 18, 19, 32, 40, 41.
Oriented paths, 18, 19, 41.
  spanning, 40.
Orthogonal arrays, 37, 139.
  generalized, 137–138.
Orthogonal DNFs, 84–85, 92.
Orthogonal latin squares, 3–8, 36–38.
Orthogonal strings, 37.
Orthogonal vectors, 34, 37.
Otter theorem-proving program, 167.
Out-degree of a vertex, 18, 19, 21, 40, 41, 43.
Overlapping subtrees, 97.
Ozanam, Jacques, 3, 7, 9.

$P = NP(?)$, 55.
P_n (path of order n), 13, 28, 39.
$\vec{P_n}$ (oriented path of order n), 18, 41.
Paige, Lowell J., 5–7.
Palindromes, 38, 135, 163.
Palluel, François Cretté de, 8.
Papadimitriou, Christos Harilaos (Παπαδημητρίου, Χρίστος Χαριλάου), ix.
Parallel addition, 108, 127–128, 189.
Parallel computation, vii, 91, 108, 124.
Parallel edges of a multigraph, 19, 41.
Parallel lines, 37.
Parity bits, 38.
Parity function, 51, 77, 94–95, 98, 105, 131, 132, 184.
Parker, Ernest Tilden, 5–7, 136.
Partial cubes, 90.
Partial functions, 113–114, 131.
Partially symmetric functions, 95.
Partitions of an integer, 25, 30, 149.
parts graphs, 25.
Pascal, Blaise, iii.
Patashnik, Oren, 203.
Patents, 112.
Paterson, Michael Stewart, 126, 127.
Path graph P_n, 13, 28, 39, 157.
Paths in a graph, 13, 41.
  oriented, 18, 19, 41.
  shortest, viii, 12, 16, 32, 66.
Patterson, Nicholas James, 180.
Paul, Wolfgang Jakob, 131.
Pehoushek, Joseph Daniel, 87.

Peirce, Charles Santiago Sanders, 48, 50, 53, 161.
Perfect shuffles, 38.
Permanent of a matrix, 40, 119, 125.
Permutation digraphs, 40.
Permutation matrices, 20, 143.
Perrin, François Olivier Raoul, numbers, 157.
Petersen, Julius Peter Christian, 5, 14, 145.
  graph, 14, 15, 25, 39, 42, 44, 45, 145.
Phi (ϕ), 134.
  as source of "random" data, 136.
Philo of Megara (= Philo the Dialectician, Φίλων ὁ Μεγαρίτης), 48.
Pi (π), as source of "random" data, 76, 98, 118, 128, 136, *see also* Pi function.
Pi function, 52, 54, 80–82, 118, 125, 130, 181.
Pigeonhole principle, 85, 199.
Pippenger, Nicholas John, 177, 192.
Pitts, Walter Harry, 75.
Pixels, 24, 31.
Planar graphs, 14, 15, 17, 24, 39, 44, 141, 147.
Planar Langford pairings, 36.
plane_lisa graphs, 24, 31.
plane_miles graphs, 24, 31.
PLAs, 53.
Plass, Michael Frederick, 87.
Playing cards, 3–4.
Poetry, 140.
Pohl, Ira Sheldon, 3.
Poirot, Hercule, 18–19.
Polignac, Camille Armand Jules Marie de, 15.
Polish notation, 63.
Pólya, György (= George), 18.
Polyhedral combinatorics, vii.
Polynomials, *see* Availability polynomials, Integer multilinear representation, Interpolating polynomials, Multilinear representation of a Boolean function.
Portable programs, 22.
Positive Boolean functions, 55, 156, *see* Monotone Boolean functions.
Positive normal form, *see* Multilinear representation of a Boolean function.
Positive threshold functions, 75–76, 92.
Posner, Edward Charles, 139.
Post, Emil Leon, 63, 68, 165.
Postal codes, 15, 40.
Pratt, Vaughan Ronald, 125.
Precedence of operators, 51.
Prefix problem, 127–128, 132.
Prefixes of strings, 132.
Preparata, Franco Paolo, 187.
Prime clauses, 54, 95, 129.
Prime forms, 54, 64, 71, 81.
Prime graphs, 28.
Prime implicants, 54, 64, 71, 82, 89, 94, 95, 129, 197, 200.
  of a majority function, 92, 172.
Prime-number function, 110, 129.
Priority encoders, 127.
Product-of-sums expression, *see* Conjunctive normal form.
Products of digraphs and multigraphs, 146.
Products of graphs, 27–28, 42–44.
Programmable logic arrays, 53.
Programming languages, 58.
Projection functions, 49, 63, 80, 154, 169.
Projections in a median algebra, 67.
Projective planes, finite, 138, 149, 200.
Prolog language, 57.
Proportional graphs, 46.
Provan, John Scott, 161.
Pseudorandom numbers, 12, 25.
Pun resisted, 62.
Pure majority functions, 76, 169.
Putzolu, Gianfranco, 160.
Puzzles, 1, 3, 7–9, 15, 135, 141.
Pyramids, tetrahedral, 89.

Quantified formulas, 87, 123–124, 198.
Queen moves on a chessboard, 26, 44.
Queues, 162.
Quick, Jonathan Horatio, 81, 162–163.
Quine, Willard Van Orman, 54, 55, 82, 159.
Quorums, 88.

r-closed graphs, 133.
r-families of edges, 133.
r-uniform hypergraphs, 32.
Rado, Richard, 174.
RainBones puzzle, 135.
raman graphs, 24, 151.
Ramanujan Iyengar, Srinivasa (ஸ்ரீனிவாஸராமானுஜன் ஐயங்கார்), graphs, 24, 151.
Random Boolean functions, 56, 83, 160.
random_graph graphs, 25, 41.
Random graphs, 25, 41, 46.
Random number generation, 96.
Random walks, 45.
Randomization, 118, 130.
Rao, Calyampudi Radhakrishna (కల్యంపూడి రాధాకృష్ణ రావు), 137–138.
Razborov, Alexander Alexandrovich (Разборов, Александр Александрович), 125.
Real numbers, 91.
  extended, 63.
Reckhow, Robert Allen, 162.
Recreations, 1, 3, 7–9, 15, 135, 141.
Recurrence relations, 71, 157, 185, 187.
  binary, 108, 109, 126, 168, 186.
Recursive procedures, 70, 82, 181, 185.

Red'kin, Nikolai Petrovich (Редькин, Николай Петрович), 107, 122, 131, 196.
Rédei, László, 142.
Reduced median sets, 72, 91.
Redundant coordinates, 72.
Redundant implicants, 94, 158, 160.
Registers, 101–103, 126.
Regular Boolean functions, 93, 178.
Regular graphs, 14, 24–25, 33, 40–44.
Reliability polynomials, 80, 81, 84, 93, 155.
Renaming (selectively complementing) Boolean variables, 87, 156, 163.
Representation of graphs and digraphs, 19–22.
Resolution principle, 159.
Resolvents, 159.
Restriction of a graph, 13.
Retracts, 74, 91.
Retraction mappings, 74, 91.
Right complementation ($\bar{R}$), 49.
Right projection (R), 49, 63, 154.
Ring sum expansion, *see* Multilinear representation of a Boolean function.
Ringel, Gerhard, 145.
Riordan, John, 190.
risc graphs, 31–32.
RISC: Reduced Instruction Set Computer, 32.
Rivest, Ronald Linn, 56.
Robertson, George Neil, 17.
Robinson, John Alan, 159.
roget graphs, 23, 41.
Roget, John Lewis, 23.
Roget, Peter Mark, 9, 23.
Rook moves on a chessboard, 26, 41.
Rookwise connected pixels, 24.
Rosa, Alexander, 136.
Rota, Gian-Carlo, 176.
Roth, John Paul, 159.
Rothaus, Oscar Seymour, 180.
Ruciński, Andrzej, 152.
Runs of 0s or 1s, 169.
Ruskey, Frank, 129.
Ryser, Herbert John, 37, 137.

S-boxes, 129.
$S_{k_1,\ldots,k_t}(x)$ notation, 77, *see* Symmetric functions.
Sachs, Horst, 145.
Saks, Michael Ezra, 168.
Samson, Edward Walter, 156, 159.
Sanders, Daniel Preston, 17.
Sangiovanni-Vincentelli, Alberto Luigi, 122.
Sartena, Christian, 154.
Sasaki, Fukashi (佐々木不可止), 77.
Satisfiability problem, 55–62, 85–87.
  for Horn clauses, 60, 85–86.
  for Krom clauses, 57, 60–62, 72, 85–86.
Satisfiable Boolean formulas, 55.
Saturating subtraction ($\dot{-}$), 49, 84.
Sauerhoff, Martin, 153.
Sauveur, Joseph, 136.
Savage, John Edmund, 190.
Scandalous fact, 26.
Schaefer, Thomas Jerome, 72.
Scheduling, 60–62, 86.
Schensted, Craige Eugene (= Ea Ea), 87–89, 166.
Schmitt, Peter Hans, 165.
Schnorr, Claus-Peter, 131.
Schröder, Friedrich Wilhelm Karl Ernst, 80, 155.
Schroeppel, Richard Crabtree, 126.
Schumacher, Heinrich Christian, 5, 17.
Scrabble®, 10.
Search trees, 6.
Second-order logic, 124.
Seed value for pseudorandom numbers, 12, 25.
Selection function, *see* Mux.
Self-complementary graphs, 42, 43, 153.
Self-converse graphs, 145.
Self-dual Boolean functions, 63, 79, 81, 92, 95.
  monotone, 63–64, 70, 79, 87–89, 92–93, 133.
  threshold, 79, 92.
Self-dualization, 92.
Self-loops, 13, 18, 19, 41, 148.
Self-organizing data structures, 168.
Self-reference, 124, 213.
Semidefinite programming, vii.
Separable functions, *see* Threshold functions.
Sequential algorithms, vii.
Sequential allocation, 21.
Series-parallel switching networks, 190.
Server locations, 91.
Set systems, 32, *see* Hypergraphs.
Seven-segment display, 112–114, 129.
Seymour, Paul Douglas, 17.
SGB, 9, *see* Stanford GraphBase.
Shadows of bit codes, 84.
Shakespeare (= Shakspere), William, 1.
Shannon, Claude Elwood, Jr., 47, 110, 179, 189, 190.
Sheehan, John, 151.
Sheep, 8, 38.
Sheffer, Henry Maurice, 50, 80.
Shelling a monotone Boolean function, 84–85, 157, 161, 171.
Shen, Vincent Yun-Shen (沈運申), 118, 120, 131.
Shmulevich, Ilya Vladimir (Шмулевич, Илья Владимирович), 178, 179.
Sholander, Marlow Canon, 89, 167.
Sholomov, Lev Abramovich (Шоломов, Лев Абрамович), 129, 194.

Shortest normal forms, 55, 82, 83, 95, 156.
Shortest paths in a graph, viii, 12, 16, 32, 66.
Shrikhande, Sharadchandra Shankar (शरदचन्द्र शंकर श्रीखंडे), 5.
Shrinking an edge, 141.
Sideways addition function (νx), 77, 90, 108, 126, 158, 185, 187.
Signed permutations, 178.
Simple digraphs, 18, 19, 40, 43, 145, 146.
Simple games, 169.
Simple graphs, *see* Graphs.
simplex graphs, 25.
Simpson, James Edward, 134.
Single-faced functions, *see* Canalizing functions.
Singleton, Robert Richmond, 143.
Singmaster, David Breyer, 135.
Sink vertex, 18.
Sinks, 62.
Size of a digraph, 18.
Size of a graph, 13, 44.
Sklansky, Jack, 187.
Skolem, Albert Thoralf, ix, 8, 36, 134, 135.
Sloan, Robert Hal, 159.
Sloane, Neil James Alexander, 138.
Slutzki, Giora (גיורא סלוצקי), 69.
`smile`, 11, 16, 24, 39.
Smith, Henry John Stephen, normal form, 152.
Smith, John Lynn, 188.
Smith, Mark Andrew, 96.
Snir, Marc (מרק שניר), 127, 132.
Sorting networks, 91, 127, 186.
Source vertex, 18.
Sources, 62.
Spanning subgraphs, 13, 15, 18, 39.
Sparse graphs, 20, 23.
Spectrum of an irrational number, 134.
Spheres, vii.
Spira, Philip Martin, 125.
Stability number $\alpha(H)$ of a graph or hypergraph, 35.
Stable sets, 34, *see* Independent vertices.
Stack structures, 23, 41, 59, 158, 162.
Standard fields in SGB format, 21.
Stanford Cardinal, 31.
Stanford GraphBase, ii, iv, viii, 9–12, 20, 23–26, 31, 62, 188.
  complete guide to, 32.
  format for digraphs and graphs, 21–22, 41.
Stanley, Richard Peter, 13, 173.
Star graphs, 17.
Stein, Sherman Kopald, 192.
Steiner, Jacob, 17.
  trees, vii, 17.
  triple systems, 8.
Stockmeyer, Larry Joseph, viii, 123, 124, 132, 185, 198.
Stone representation, 88–89.
Storage access function, 109.
Strahler, Arthur Newell, 140.
Straight insertion sorting, 142.
Straight-line computations, *see* Boolean chains.
Strassen, Volker, 200.
Strong components: Strongly connected components, 40, 61–62, 86, 142, 164, 195.
Strong product of graphs, 28, 42–44, 146, 151.
Stufken, John, 138.
Subcubes, 54, 82–84, 129, 155, 160, 161, 176, 190.
  maximal, 54–55, 82–83.
Subgraphs, 13, 17.
  of a hypercube, 90.
Subramani, Krishnamurthy (கிருஷ்ணமூர்த்தி சுப்ரமணி), 165.
Subramanian, Ashok, 170.
subsets graphs, 25.
Subwords, 12.
Suffixes of strings, 132.
Sum-of-products expression, *see* Disjunctive normal form.
SWAC computer, 5–6.
Swift, Jonathan Dean, 47.
Symmetric block designs, 179.
Symmetric Boolean functions, 77–79, 94.
Symmetric functions, 98–99, 104–106, 108, 109, 116, 126, 131, 132, 184, 185.
Symmetric matrices, 40, 44.
Symmetries of a graph, 14–15, 39, 45, 147, 152–153.
Syntax, context-free, 58.
Szele, Tibor, 142.
Szemerédi, Endre, 91.
Szörényi, Balázs, 159.

Tableaux, 29–30, 43.
Tables of Boolean function counts, 79.
Tags, 82.
Takahasi, Hidetosi (高橋秀俊), 171.
Takasu, Satoru (高須達), 171.
TAKE_RISC program, 32.
Tannenbaum, Meyer, 175.
Tarjan, Robert Endre, 1, 62, 87, 195.
Tarry, Gaston, 5.
Tautology ($\top$), 49, 161.
Taylor, Brook, series, 13.
Tensor product of graphs, *see* Direct product of graphs.
Terminology, 13, 156.
Ternary Boolean functions, 104–105, 200.
Ternary Boolean operations, 63, 156.
  table, 78.
Tetrahedra, 25.
Tetrahedral pyramids, 88–89.
TeX, 12.

Theorem proving, 59, 167.
Theory meets practice, vii, 13.
Thomas, Herbert Christopher (= Ivo), 49.
Thomas, Robin, 17.
Thoreau, David Henry (= Henry David), 96.
Three-variable functions, 63, 104–105, 156.
 table, 78.
Threshold functions, 75–77, 79, 92–95, 186.
 of threshold functions, 77, 92, 172.
Thue, Axel, ix.
Tic-tac-toe, 114–117, 129–130, 192.
Tightly colorable graphs, 44.
Tilings of the plane, 45.
Tip of an arc, 18.
TIP(a) (final vertex), 21.
Tison, Pierre Louis Joseph, 159.
Toda, Iwao (戸田巌), 171.
Todorov, Dobromir Todorov (Тодоров, Добромир Тодоров), 137.
Tolstoy, Leo Nikolaevich, (Толстой, Лев Николаевич), 9.
Tomlin, Mary Jean (= Lily), 60, 62, 86.
Tompkins, Charles Brown, 5–7.
Top-down synthesis, 103–105, 126, 183.
Topological sorting, 60, 73, 85, 97.
Toruses, 28, 41.
 generalized, 45–46.
 kingwise, 44.
Tournaments, 40.
 transitive, 18, 27, 40, 41.
Transitive laws, 138.
Translation, tiling by, 45–46.
Transposing a matrix, 20, 33, 137.
Transversals of a latin square, 6–7, 37, 136.
Traveling Salesrep Problem, viii.
Trees, 67, 81, 85, 91.
Triangle function, 133.
Triangle inequality, 16, 19.
Triangular grids, 25, 88, 145.
Tries, 38–39.
Triple systems, 8, 32, 44.
Trivalent graphs, 14, 39, 151.
Trivial functions, 49, 63, 64, 184.
Truth, 47, 63, 79.
Truth tables, 47, 49, 51–53, 71, 94–96, 98, 100, 102, 105, 154, 189.
 in hexadecimal notation, 105, 132, 191.
 of partial functions, 114–116.
 two-dimensional, 110–111, 117–121, 192, 194.
Tsuboi, Teiichi (坪井禎一), 172, 173, 178.
Tukey, John Wilder, 47.
Turán, György, 159.
Tweedledee, 48, 79.
Two-variable functions, 47–50, 79–80.
 table, 49.

Uhlig, Dietmar, 131.
Ulyanov, Vladimir Ilyich (Ульянов, Владимир Ильич), 86.
Unary operator: A function of one variable.
Unate functions, 156.
Union-find algorithm, 69.
Union of graphs, 26, *see also* Direct sum of graphs.
United States of America, contiguous, 15, 34, 39–40.
UNIVAC 1206 Military Computer, 5.
Universal algebras, 155.
Universal quantifiers, 87.
Universities, 31.
Unlabeled graphs, 14, 152.
Unsolvable problems, 164.
Upper bounds on combinational complexity, 110–112, 129, *see also* Footprints.
Utilities, 141.
Utility fields in SGB format, 21, 144.

Vacuum-tube circuits, 104, 126.
Valency, *see* Degree of a vertex.
Validity $(\top)$, 49.
Variance, 152.
Vector-valued Boolean functions, 107, 121.
Vectors, 54.
Veroff, Robert Louis, 167.
Vertex connectivity, 148.
Vertex covers, 157.
 minimum, 34–35, 44.
Vertex variables, 21, 23.
VERTICES(g) (the first vertex node), 22, 143.
Vesztergombi, Katalin, 149.
Vikulin, Anatoly Petrovich (Викулин, Анатолий Петрович), 176.
Vinci, Leonardo da, 9, 24.
von Neumann, John (= Margittai Neumann János), 169.
Vowels, 38.
Vũ, Văn Hà, 175.

W_n (wheel graph of order n), 42, 46, 145, 153.
Wagnalls, Adam Willis, 48.
Wagner, Eric Gerhardt, 159.
Wakerly, John Francis, 194.
Walks in a graph, 19–20, 40, 45.
Walter of Burley (= Burleigh = Gualterus Burleus), 51.
Wang, Da-Lun (王大倫), 148.
Wang, Shinmin Patrick (王新民), 137.
Watkins, John Jaeger, 151.
Weak second-order logic, 123–124, 132.
Weber, Karl, 160.
Wegener, Ingo Werner, 124, 153, 185.
Weichsel, Paul Morris, 146.
Weinberger, Arnold, 188.
Weiner, Peter Gallegos, 118, 120, 131.
Weisner, Louis, 136.
Well-balanced Langford pairings, 2–3, 36.
Wernicke, August Ludwig Paul, 5.

Wheel graphs, 42, 46, 145, 153.
Width of a Langford pairing, 3, 135.
Wiedemann, Douglas Henry, 73, 180.
William of Ockham (= Guilielmus ab Occam), 51.
Williams, Robin McLaurim, 60, 62, 163.
Wilson, Richard Michael, 137.
Wilson, Robin James, iv, 15, 42, 151.
Winder, Robert Owen, 64, 173, 200.
Witness bits, 139.
Wood, Frank W., 112.
Wool, Avishai (אבישי וול), 174.
Wong, Chak-Kuen (黃澤權), 152.
Word cubes, 11, 39.
Word ladders, 11–12, 32.
Word problems, 68.
Word squares, 11, 38.
words graphs, 12–13, 31, 39, 42.
WORDS(n), the n most common five-letter words of English, 10–12.
Working units, 80.
Wraparound, 41.
WS1S, 124.

XOR (exclusive or, ⊕), 49–52, 77, 80, 81, 154, 172.
  bitwise, 90, 158.
XOR gates (⊕), 32, 33, 97.

Y functions, 88–89, 92.
Yablonsky, Sergei Vsevolodovich (Яблонский, Сергей Всеволодович), 177.

ZDDs: Zero-suppressed decision diagrams, 166.
Zehfuss, Johann Georg, 146.
Zero-one principle, 68.
Zhao, Xishun (赵希顺), 165.
Zhegalkin (= Gégalkine), Ivan Ivanovich (Жегалкин, Иван Иванович), 51, 155.
Zigzag paths, 177.
Zito, Jennifer Snyder, 152.
Zuev, Yuri Anatol'evich (Зуев, Юрий Анатольевич), 178.
Zuse, Konrad, 189.
Zykov, Aleksander Aleksandrovich (Зыков, Александр Александрович), 26.

计算机程序设计艺术

第4卷　第0册

组合算法与布尔函数概论

PDG

译 者 序

本书是克努特（Knuth，中文名为高德纳，亦翻译为克鲁特）巨著《计算机程序设计艺术》第4卷第0册的双语版（中文版），自从第4卷第2、3、4册的双语版出版以来，众多读者翘首以待第4卷前面各册双语版的出版已经有一段时日了，希望本书不会让大家失望。

译者最早是在20世纪80年代初在图书馆看到标为“克鲁特著，管纪文、苏运霖译”的《计算机程序设计技巧》多卷本的，那时译者在浙江大学上大学，正学习Aho等著的《计算机算法的设计与分析》（由译者翻译的该书中文版已由机械工业出版社出版），由于当时用的是原版书籍，许多概念一头雾水，正是该书帮我澄清了很多概念，打发了许多时间，也培养了我阅读的兴趣。

后来写些小论文，经常使用LaTeX这个软件，惊叹于它的美丽，看了软件的说明，知道是从事分布式计算研究的著名学者兰伯特（Leslie Lamport）在克努特的TeX上构建的宏集，也了解了克努特暂时中断巨著撰写的原因，明白了“工欲善其事，必先利其器”的道理。

事实上，克努特从来就没有中断过巨著的写作计划，几十年来他孜孜不倦地搜集素材，了解计算机科学各方面的进展，我们期望他身体康健，厚积博发（这里没有写错）。

2000年我在哈佛大学进修的时候，研究程序设计语言的Ramsey教授邀请克努特到哈佛来做个讲座，我才遇到他本人，看到的是无腿水晶眼镜后面那双睿智的眼睛，听到的是娓娓道来的各种典故新说，他介绍的是他自行设计的MMIX计算机，虽然一开始克努特就说道，我要讲到没有一个人想听的时候为止。可教室的十多个人都倾心听讲并不时发问，直到晌午，大家吃了主持者叫来的外卖比萨，还孜孜不倦地继续讨论着他设计的新千年RISC机器。

借此机会再谈一下本书，本书是第4卷第0册，介绍的是组合搜索的一些预备知识，要阅读第4卷后面各册的读者最好先读一下本书。另外，《计算机程序设计艺术》一书的很多习题富有挑战性，按照作者的标记，0是最容易的题目，45是已解决的难题，如费马大定理等，而大于45的基本是未知答案的难题，这也给大家设立算法研究目标提供了参考。

本书的习题和解答由杜思奇、曾慧清、张王朋和恽筱源初译，最后由黄林鹏整理。本书的翻译还得到孙永强教授和梅宏、童维勤、陆朝俊、陈翌佳等的帮助，在此表示真挚的谢意。

对于此书的翻译，说实在话，一是英语非译者母语，新翻译卷册的质量恐难延续苏运霖教授翻译的广受好评的三卷；二是整书涉及内容非常广泛，译者肯定力有不逮。无奈出版社多次提及，译者只能勉强为之。错误难免，如果读者发现敬请发到lphuang@sjtu.edu.cn，以便再版修改，不胜感激。

黄林鹏
于上海交通大学
2009年12月

PDG

前　言

将所有好素材都放进一本书显然不可能，对一个主题的某个方面进行适当广泛的讨论甚至也可能导致篇幅的增长失去控制。

——杰拉尔德·弗兰德（Gerald B. Folland）编辑之角（2005）

著书立说最重要之事便是决定置何于卷首。

——布莱斯·帕斯卡尔（Blaise Pascal）《思想录》740（c.1660）

本书是《计算机程序设计艺术，第4卷：组合算法》的第0册。如本书第1卷第1册的前言所述，本书以这种分册的非完整形式出版是因为第4卷的写作将延续多年，而我却迫不及待地等待人们阅读我业已完成的部分内容并提供他们宝贵的反馈意见。

从整卷的情况看，本册是组合算法这一宏大篇章的起始。如果我身体状况许可，那么第7章会是《计算机程序设计艺术》一书最长的一章，其最终将划归为至少3个分卷（4A、4B和4C）。和篇幅次长的第5章类似，出现在主要内容之前的是介绍性的材料，其中包含了数十个习题。即将开始的漫漫征途需要备好充足的重要补给。另外，由于第6章的撰写和出版已有30多年了，我需要使第6章过渡到新一章的过程中出现的冲突尽可能少。

第7章以7.1节，即0和1，作为合适的起点，虽然层次不同，也是引论性的内容，在此我们深入探究作为计算机基础的布尔函数相关的所有重要问题。其中，7.1.1节为以后要建立的重要上层建筑打下坚实的理论和实践基础，7.1.2节则考虑如何最有效地计算一个布尔函数的值的问题。

7.1节的剩余部分7.1.3节和7.1.4节将作为第4卷第1册出版。接着的是7.2节，生成所有的可能性。对应于7.2.1节的分册已经出版，7.2.2节将从总体上对回溯问题进行研究。如果一切顺利，本书的篇章将继续延续，而目前规划的整个第7章的轮廓，可参见封底列出的TAOCP网址。

这些介绍性小节的习题数比我原先规划的要多出一倍。事实上，本册的习题总数是难以置信的336，但其中许多十分简单，目的是强化读者对基本概念的理解，或对斯坦福图库（The Stanford GraphBase）的了解。而其他习题真的是很有吸引力的，很有必要在这里出现。不管你相信与否，我拒绝了许多比这里出现的练习更有诱惑力的习题。

感谢Robert W. Floyd，1977年他应我之邀阅读了本书的草稿并给出了数十个极有价值的建设性意见。感谢开放大学的Robin Wilson，他仔细阅读了本书并提出了许多详细的建议。感谢数以百计的读者，他们针对本书草稿提出了许多极具价值的反馈意见并发布于互联网上。

对于本书出现的每一个错误，不管是印刷上的、技术上的还是与历史相关的，我都将乐意奉上2.56美元的酬谢给第一个报告的读者。如果你发现了忘了放进索引的条目，也可得到同样的酬谢。每个对行文进行改进的建议的酬谢是32美分（而且，如果你发现了一个习题更好的解，那么你得到的将远非金钱，作为荣誉性的奖励，我会在最后的出版物中列入你的名字）。

本册所用但未加说明的记法可参见第1、2或3卷中的记法索引。索引给出了与记法相关的进一步信息之所在（你还可参见各册“记法说明”之下的内容）。当然，未来某个时日，第4卷也会有它自己的记法索引。

与尚未完成的内容间的交叉索引有时以“00”标示，这个现在不存在的值留作他日填写实际页码之用。

祝大家阅读愉快！

D. E. K.

加利福尼亚，斯坦福

2008年1月

PDG

第4卷前言

第4卷的标题是组合算法，给出标题的时候我就很想加上一个子标题：我最喜欢的程序设计类型。但我的编辑认为题目应该不带感情色彩。但事实仍是，我一直喜欢散发有组合风味的程序。

另一方面，我经常惊奇地发现，在许多人的脑海里，单词“组合”是和计算难度联系在一起的。确实，在Samuel Johnson编著的闻名的英语语言字典（1755）中，他就称其相应的名词“现通常被用于引起痛苦的场合”。同事告诉我他遇到挫折时会说“各种困境的组合打败了我”。为何会这样呢？对我而言，组合学所引发的是完全纯净的喜悦；难道对其他人，它所唤起的会是纯粹的恐慌吗？

确实，组合问题通常和巨大无比的大数相联系。Johnson的字典包括了一个引自Ephraim Chambers的说明，他说道，一个24个字符的字母表上的长度小于等于24的单词量有1 391 724 288 887 252 999 425 128 493 402 200个之多。而对应于10个字符的字母表的单词量是11 111 111 110；当字符数是5时，单词量则是3905。以此推论，当字母表中的字符个数从5到10再到24依此继续增长时，“组合爆炸”问题无疑将会发生。

在我的一生中，计算机的处理能力发生了巨大的变化。在输入这些单词时，我知道处理它们的计算机的速度比起我开始致力于此书写作时的IBM 650型计算机快了10万倍，其内存容量也大了10万倍。将来的计算机计算速度会更快，容量也会更大。这些令人惊诧的进步不会降低人们对得到组合问题答案的渴望，而恰恰相反，曾经是不可想象的计算能力的出现激起了人们的期望，刺激了大家的胃口，因为若n仅增量1，组合问题的规模就可能会增大10万倍。

组合算法可以非形式化地定义为对组合对象，如排列或图，进行高速操作的技术。我们通常尽力发现满足一定约束条件的最好的可能模式或排列方式。此类问题的数目非常巨大，编写这种程序的技艺通常特别重要又令人着迷，一个好的想法就可省去几年甚至数个世纪的计算时间。

实际上，对于组合问题，好的算法有着极大的回报，这也导致了目前该领域研究上的快速发展。许多原先被认为是不可解的难题现在被修正为容易被解决的问题，许多原先被认为是好的算法也变得更加有效了。大约从1970年开始，计算机科学家开始了解到一种被称为“弗洛伊德引理”的现象：似乎需要n^3步操作才能解决的问题实际上可在$O(n^2)$步内得到答案，似乎需要n^2步操作才能解决的问题实际上可在$O(n\log n)$步内得到答案，而$n\log n$通常可归约为$O(n)$。对于更困难的问题，运行时间也可以从$O(2^n)$降为$O(1.5^n)$再降为$O(1.3^n)$等。一些一般来说仍是困难的问题，也有很简单的重要特例。许多我曾经认为不会有答案的组合问题现在已被解决，当然，问题的突破主要是由于算法的改善而非处理器速度的改进。

到了1975年，这个领域的研究进展突飞猛进，计算机科学重要期刊上发表的很大一部分文章都是针对组合算法的。这些成果不完全是由计算机科学的核心人士做出的，许多重要的贡献是由电子工程、人工智能、运筹学、数学、物理学、统计学和其他领域的工作者做出的。正尝试着完成《计算机程序设计艺术》第4卷写作的我，感到自己似乎是坐在一个水正烧开的水壶盖上，我面临另一种类型的组合爆炸，新的思想正如泉涌般迸发。

本系列丛书诞生于1962年初，当时我写下了12章的标题临时列表。那时，仅仅是为了乐趣，我才决定将组合算法的短暂篇章包含其中。"嗨，看一下，大多数人使用计算机来处理数值，而我们能编写处理模式的程序。"在那个时代，给出所有已知的组合算法的完整描述也是容易的事。甚至在1966年，在我完成了大约3000页手写书稿的时候，在业已膨胀的篇幅之中也只有100页是关于第7章的。至于原先以为的一碟"沙拉点心"最后是如何变成一道大餐的，我却完全没有概念。

由于越来越多的人参与，起始于1975年的组合研究热仍方兴未艾。旧的思想不断被新的思想所改进，但却很少被取而代之或过时。当然，我不得不放弃我曾经的希望，即围绕整个领域撰写一本最完美、最可靠的著作，使所有的素材都井然有序，给所有希望解决组合问题的人提供一站式的购物便利。在讨论一个小的主题的时候我也几乎无法宣称"这是最后的解决方案，故事到此结束。"取而代之的是，我必须将自己局限于解释大多数目前所遇到的有效组合技术的重要基础原理。到目前为止，我为写作第4卷而搜集的素材比为撰写第1～3卷而准备的材料要多上一倍不止。

大量的内容意味着曾经规划的"第4卷"实际上会变成多个自然卷。你现在看的是4A卷，如果我身体健康，4B卷和4C卷也会在某个时日出现，而且（谁知道呢？）也可能会有4D卷、4E卷……但肯定不会有4Z卷。

我的计划是系统地浏览我从1962年就开始积累的素材，尽我所能，告诉大家我认为大家还期待着被告知的事情，我并不追求完美，但我想给那些对组合思想作出贡献的所有先驱者们应该享有的荣誉，因此我不会吝啬任何历史细节。此外，对于那些我认为50年之后也是重要的素材，我不会将其舍去，我会使用一到两段的篇幅优雅地对其加以解释；相反，需要冗长证明的困难问题，除了真正基本的主题之外，我就不放在本书进行讨论了。

好吧，组合算法的研究范围是如此广泛，显然我是无法涉足所有领域的。哪些重要的东西会被略掉呢？我认为我最大的盲点是几何。因为，比起空间对象，我能更好地想象并操作代数公式。因此在本书中，我不想去尝试处理和计算几何相关的组合问题，如球的紧密堆积问题、n维欧几里得空间中数据点的聚类问题，甚至平面上的Steiner树问题等。更重要的是，在本书我倾向于不涉及多面体组合学问题和基于线性规划、整数规划或半定规划的组合学问题。这些主题在许多书籍已充分讨论，但它们依赖于几何直觉，而对我来说，纯粹的组合学问题则更容易理解。

我必须坦白，我对那些仅仅在渐近意义上是有效的算法存有偏见，这些算法

直到问题的规模超出了宇宙的边界才有可能呈现出性能上的优越性。目前发表的许多文章都致力于此类算法的研究。我能理解挑战终极极限的诱惑以及建立学术威望对学者的吸引力，但在《计算机程序设计艺术》一书中，对那些我从未考虑过在实际编程中自身会采用的方法，我倾向于只给出简短的说明（当然，对此也有例外，特别是一些涉及核心主题的基本概念，还有一些不切实际但漂亮且/或富有洞察力的方法也包含在书中，其他则是用于说明什么是不该做的指导性例子）。

而且，如本书早期出版的其他各卷，本书几乎完全专注于串行算法的讨论，这是有意而为之的。尽管计算机能越来越好地并行执行任务，但我无法判断，涉及并行的这些思想从现在开始的5年或10年之后是否还有用，因此我乐意将此问题留给那些比我睿智的人。串行方法已经挑战了我的智力极限，我不想去辨明未来的精明程序设计者想要知道什么。

当规划如何将素材呈现给读者时，我需要作出的主要决定是应该按问题还是按技术进行内容的组织。例如，第3卷第5章，主要讨论的是一个单独的问题，即如何将数据按序进行排列，对于问题的不同方面，我们应用了几十种不同的技术。与之相反，组合算法涉及许多问题，而可施用的技术种类却很少。最后我认为混合策略应比任何纯粹的策略都要好。因此，你会看到，7.3节处理寻找最短路径问题，7.4.1节讨论图的连通性问题，而其他章节则涉及一些基本技术的讨论，如布尔代数（7.1节）、回溯技术（7.2节）、拟阵理论（7.6节）和动态规划（7.7节）等。著名的旅行商问题虽然没有属于自己的独立小节，但可被不同的方法解决，因此会在不同的地方出现多次。

前面提及，在组合计算的研究上已经取得了巨大的进展，但这并非意味着所有的组合问题都已被解决。当一个计算机程序的运行时间变得无法控制时，程序设计者不要期望在本书能找到他所需要的灵丹妙药。虽然本书描述的方法通常会比程序设计者前期自行尝试的方法快上很多，但我们必须面对现实：组合问题很快就会变得极大。我们甚至可以严格证明，某些小而常见的问题，尽管理论上可解，但在现实世界中是不可能得到解的（参见7.1.2节中的Stockmeyer和Meyer定理）。对于其他一些问题，我们甚至无法证明是否会有令人满意的算法存在，但我们知道对于这些问题存在解决方法是不大可能的，因为任何一个有效算法都将导致成千其他问题的解的出现，而这些问题已经难倒了世界上最杰出的专家学者（参见7.9节NP完全问题的讨论）。

经验提示，针对新组合问题或旧组合问题的变体或特例，新的组合算法会不断涌现，而人们对于这种算法的需求也将不断增长。面对挑战，程序设计者将使计算机程序设计艺术不断跨越新的巅峰。但是，今天的方法很可能仍有价值。

尽管和第1～3卷讨论的主题常有联系，本书的大部分内容是自足的。在前面几卷中，与机器语言相关的底层细节被广泛涉及，但本卷中的算法则独立于任何机器，通常只在抽象层次上进行阐述。然而，组合编程的某些方面严重依赖于底层细节，这在前面是未曾出现过的。在这种情况下，本书都是基于MMIX计算机进行讨论的。MMIX取代了在第一卷早期版本中所定义的MIX机器。关于MMIX机

器的细节可参阅《计算机程序设计艺术》第1卷第1册的补充材料，读者也可从因特网下载这些材料，同时可下载的还有MMIX汇编程序和仿真器。

另一个可下载的资源是称为斯坦福图库（The Stanford GraphBase）的程序集和数据集，它们被本书例子所大量引用。我认为学习组合算法最有效、最有意思的方式就是使用该图库，因此我在此鼓励读者充分使用它们。

顺便提一下，在撰写作为第7章开场白的介绍性内容时，我很高兴能自然地提及我的博士论文导师Marshall Hall, Jr.（1910—1990）、Hall的论文导师Oystein Ore（1899—1968）和Ore的论文导师Thoralf Skolem（1887—1963）的一些工作。而Skolem的导师Axel Thue（1863—1922）的工作则已在第3卷第6章提及。

非常感谢数以百计的读者，他们帮助我在本卷早期的版本中发现了众多的错误，这些版本最初发布于因特网后来再以纸质分册的形式印刷出版。我恐怕还有其他错误仍然潜伏于字里行间，我希望能尽可能地将所有错误予以改正。因此，对于每个技术上的、印刷上的还是历史上的错误的第一个发现者，我都很乐意给予2.56美元的酬谢。在封底的TAOCP网址，你可找到当前已被报告的所有错误的列表。

D. E. K.
加利福尼亚，斯坦福
2008年4月

尽管，依我看来，错误是我朋友造成的，但自然地，
我要对所有出现的错误负责。
——奎斯托·帕察措格卢（Christos H. Papadimitriou），
《计算复杂性》(1995)

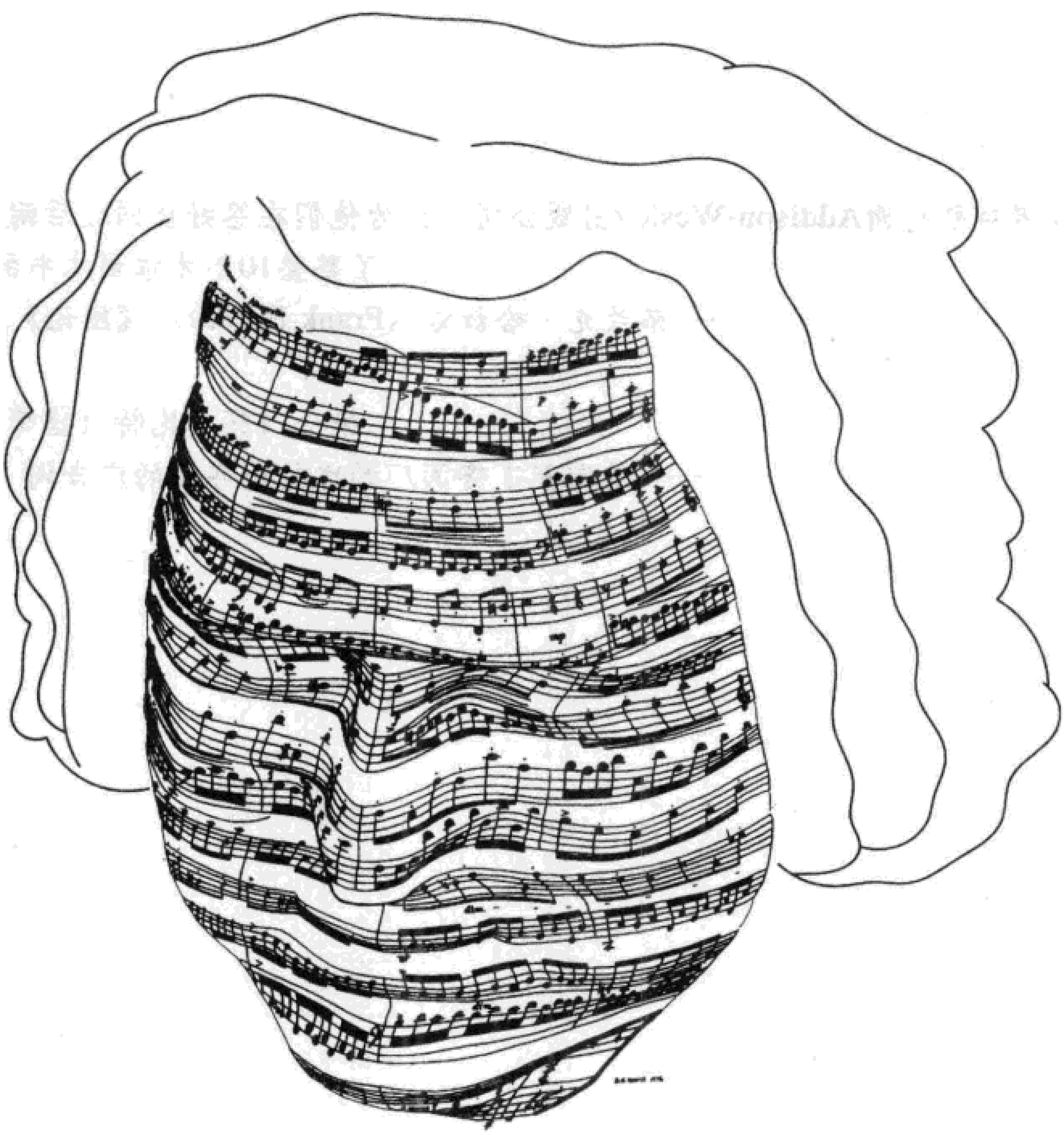
关于引用 对IEEE Transactions杂志的引用由一个用于标识杂志类型的黑体字符后接杂志卷号构成，如“IEEE Trans. C-35”表示IEEE Transactions on Computers第35卷。IEEE不再使用这些便利的字符编码，但对这些编码的辨认并不太难：如“EC”曾用于表示“电子计算机”(Electronic Computers)，“IT”表示“信息理论”(Information Theory)，“SE”表示“软件工程”(Software Engineering)，“SP”表示“信号处理”(Signal Processing)，“CAD”表示“集成电路系统的计算机辅助设计”(Computer-Aided Design of Integrated Circuits and Systems）等。

作者特别感谢Addison-Wesley出版公司，因为他们在签好合同之后耐心等待了整整10年才拿到本书的手稿。

——弗兰克·哈拉雷（Frank Harary），《图论》(1968)

来一（杯）比特（堡啤酒）！

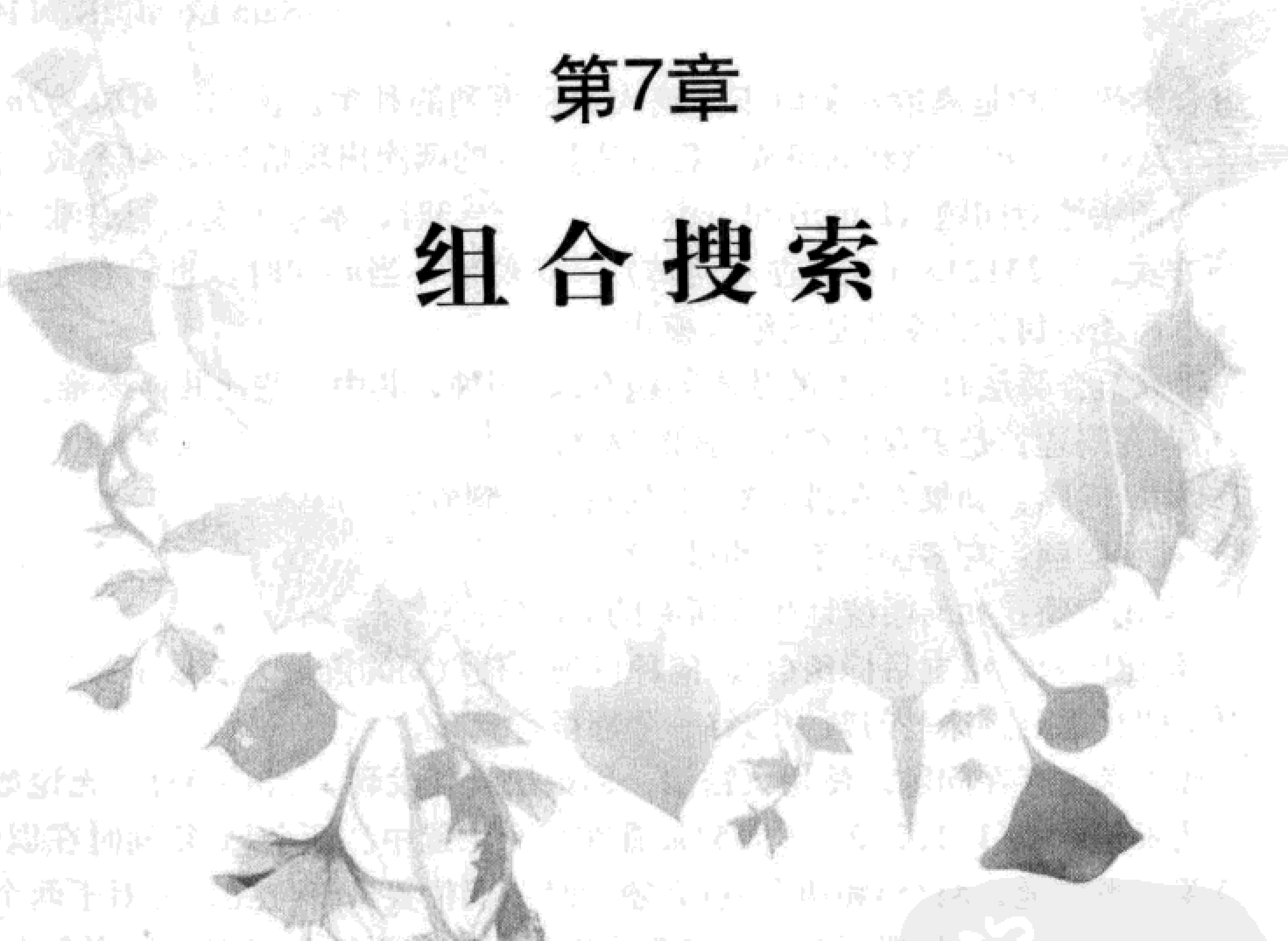
——德国宾丁啤酒厂的比特堡啤酒的广告词（1951）



Hommage à Bach.

第7章

组合搜索



（就像藏在两桶砻糠里的两粒麦子，）你必须费去整天工夫才能够把它们找到，可是找到它们以后，你会觉得费这么多气力找它们出来，是一点不值得的。

——巴萨尼奥，《威尼斯商人》

（第一幕，第一场，第117行）

在众杂人群间尔虞我诈的权益争逐之中，各种错综复杂的事情都有可能发生，其中有许多小问题，它们虽看起来既惊人又奇怪，却不是犯罪线索。

——福尔摩斯，《蓝宝石案》（1892）

组合算法涉及的领域是如此广阔，一篇论文或一册书不可能将其完全论述。

——Robert E. Tarjan（1976）

人有形形色色，但在我的印象中成功者只是那些天生就能看穿迷局的人。人生就是一个谜局，命运与归途相伴，活着就是为了求证生命的真谛。

——Sam Loyd, JR.（1927）

组合学研究的是离散对象以不同模式类别排列的种类。例如，对象为$2n$个数$\{1, 1, 2, 2, \cdots, n, n\}$，欲将其排成一行，使数字$k$的两次出现恰好相隔$k$个数，这个问题称为朗福德对问题（Langford pair）[⊖]。当$n=3$时，本质上说，只有唯一的一种排列方式，即231213（及其左右逆转）；类似地，当$n=4$时，也只有唯一的一个解。下面还会讨论许多其他的组合模式。

在研究组合算法时，典型的基本问题有以下5种，其中一些比其他要难。

1）存在问题：是否存在符合模式的排列X？

2）构造问题：如果存在排列X，如何快速找到它？

3）枚举问题：有多少种不同的排列X？

4）生成问题：如何系统地生成所有的X_1，X_2，$\cdots$？

5）优化问题：给定目标函数f，何种排列可使$f(X)$的值最大或最小？

对于朗福德对，这些问题中的每一个都很有意义。

例如，考虑存在问题。使用试错法，你很快就会发现，当$n=5$时，无论如何，你都无法妥善地把$\{1, 1, 2, 2, \cdots, 5, 5\}$放置在10个位置中。两个1必须同时在偶位置或奇位置，类似地，两个3和两个5也必须同时在偶位置或奇位置；但对于两个2和两个4，必然一个在偶位置而另一个在奇位置。因此，无论如何都无法使奇位置数和偶位置数都为5。同理可以证明：当$n=6$，或更一般的情况，在$\{1, 2, \cdots, n\}$中奇数的数目是奇数时，问题无解。

换句话说，仅当$n=4m-1$或$n=4m$时，其中m为整数，朗福德对才存在。与之相反，对于不是上述形式的n，Roy O. Davies给出了放置数字的另一种优美方法（参见习题1）。

⊖ 苏格兰数学家C. Dudley Langford在观察其儿子玩耍彩色积木的过程中思考并提出了这个问题。更详细的信息可参见http://legacy.lclark.edu/~miller/langford.html。——译者注。

存有多少个本质上不同的朗福德对L_n呢？答案是很多，事实上随着n的增加，我们有：

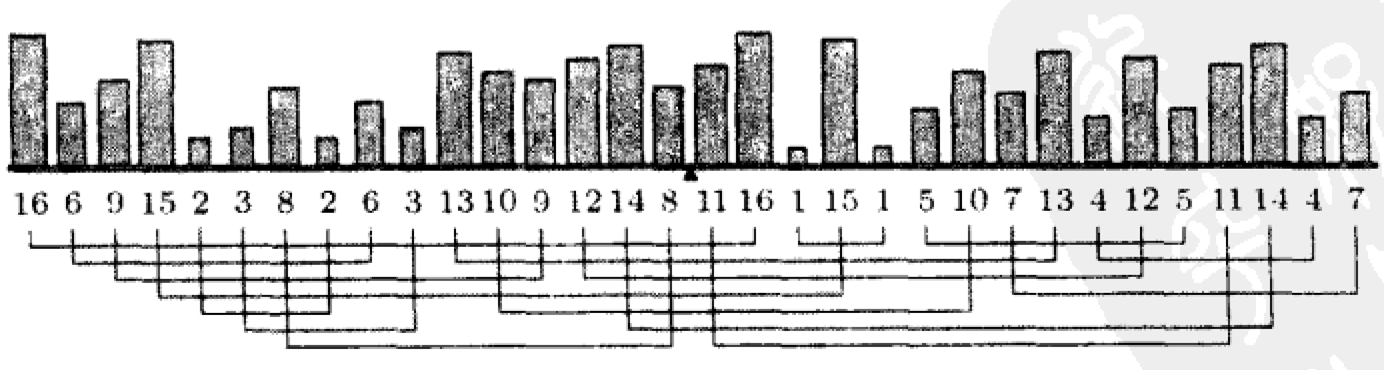
$$\begin{aligned} L_3 &= 1 & L_4 &= 1 \\ L_7 &= 26 & L_8 &= 150 \\ L_{11} &= 17\,792 & L_{12} &= 108\,144 \\ L_{15} &= 39\,809\,640 & L_{16} &= 326\,721\,800 \\ L_{19} &= 256\,814\,891\,280 & L_{20} &= 2\,636\,337\,861\,200 \\ L_{23} &= 3\,799\,455\,942\,515\,488 & L_{24} &= 46\,845\,158\,056\,515\,936 \end{aligned} \tag{1}$$

[L_{23}和L_{24}的值由M. Krajecki、C. Jaillet和A. Bui分别于2004年和2005年求得；参见《Studia Informatica Universalis》4（2005），151—190。]稍加计算凭感觉就可推得：当n非0时，L_n大约是$(4n/e^3)^{n+1/2}$阶的（参见习题5）；事实上，这个推测对于所有已知的实例都是正确的，但对此还没有一个简单的确切公式。

朗福德排列问题是组合学中称为精确覆盖问题（exact cover problem）这一一般问题类的简单特例。7.2.2.1节将讨论一种称为“跳舞链接”（dancing link）的算法，它是针对此类问题生成所有解决方案的方便之门。例如，当n=16时，为了找到每一个朗福德对排列，基于该方法我们仅需执行大约3200次内存访问。因此L_{16}的值可通过生成所有的排列再进行计数而在合理的时间内得到。

然而，要注意的是，L_{24}的值是一个庞大的数，大约为5×10^{16}，其需要大约1500MIP年才可算出（一个MIP年是一台每秒可执行1百万条指令的机器运行一年所执行的指令数，即31 556 952 000 000）。因此，L_{24}值的计算显然不应和生成所有排列的技术有关。确实，基于多项式代数，有一种非常非常快的求解L_n值的方法。习题6给出的启发式方法需要$O(4^n n)$次操作，似乎它还不是有效的，但比生成计数法在效率上高出了$\Theta((n/e^3)^{n-1/2})$阶，甚至当n=16时，速度也快了大约20倍。另一方面，L_{100}的精确值，就算计算机的速度变得越来越快，也可能永远不会被人知晓。

考虑朗福德对的各种不同的优化（optimum）问题。例如，欲对16对权重{1, 1, 2, 2, ⋯, 16, 16}进行排列，结果不仅要满足朗福德对的条件，而且还应是“均衡的”（well balanced），即该排列不会使平衡梁倾斜，当它们以恰当的顺序放置时。



(2)

换言之，就是$15.5\cdot16+14.5\cdot6+\cdots+0.5\cdot8=0.5\cdot11+\cdots+14.5\cdot4+15.5\cdot7$，对于这个特例，还有另一种类型的平衡，即$16+6+\cdots+8=11+16+\cdots+7$，进一步还有$16\cdot16+15\cdot6+\cdots+1\cdot8=1\cdot11+\cdots+15\cdot4+16\cdot7$。

另外，(2)中数的排列方式是阶为16的朗福德对中宽度（width）最小的：从上图下方的连线可以看到，在任何一点，横跨数对间的连线数不会超过7，另外还可以证明宽度为6是不可能的（参见习题7）。

设$a_1a_2\cdots a_{32}$是{1, 1, …, 16, 16}的排列，那么哪个是最小（least）平衡，即使$\sum_{k=1}^{32} ka_k$的值最大？该值又是多少？答案是最大的可能值是5 268，有12 016种，其中之一是

2 3 4 2 1 3 1 4 16 13 15 5 14 7 9 6 11 5 12 10 8 7 6 13 9 16 15 14 11 8 10 12 (3)

更有意思的问题是，若以字典序论，对于朗福德对，最大和最小的排列是什么？n=24时的答案是

{abacbdecfgdoersfpgqtuwxvjklonhmirpsjqkhltiunmwvx,
xvwsquntkigrdapaodgiknqsvxwutmrpohljcfbecbhmfejl} (4)

这里使用字母a，b，…，w，x代替数字1, 2, …, 23, 24。

本章的后续小节将陆续讨论组合优化问题。当然，我们的目标是只检查所有可能排列的一小部分而求取问题的解。

正交拉丁方阵 现在来看一下组合学发展早期的一些问题。在Jacques Ozanam的遗著《Recreations mathematiques et physiques》（Paris：1725）第4卷434页中有一个有趣的谜题："在一副普通扑克牌中取出所有的A、K、Q和J，将其排成一个方阵，每行每列都包含所有4个牌点和所有4个花色。你可以做到吗？图1是Ozanam给出的答案，该答案包含了更多的寓意，其中两个主对角线包含了所有的牌点和所有的花色（在尝试这个问题之前请不要看答案）。



图1 对所有的人头牌进行排列：4条线上的牌是不同牌点、不同花色的（构形是求解18世纪流行谜题的许多方式之一。）

到1779年，一个类似的谜题在圣·彼得堡流行，这引起了伟大的数学家欧拉的关注。这个谜题就是"设从6个不同军团各抽调6名不同军衔的军官，现欲将这36名军官排成6×6的方队，要求是每行每列都有来自不同军团、有着不同军衔的军官。怎么做？"那时没有人能给出满足条件的方阵。尽管1771年的欧拉快要失明，他还是决定通过对助手进行口述的方式来求解这个谜题。对此，他撰写了一篇专业论文[最终发表于《Verhandelingen uitgegeven door het Zeeuwsch Genootschap der Wetenschappen te Vlissingen》9（1782），85—239]，在这篇论文中，欧拉推导出有n个军衔和n个军团的类似问题的解，这里n = 1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 16, …；只有n mod 4 =2的情景使其感到困惑。

当n=2时，问题显然是无解的。在求解n=6的问题时，欧拉在检查了相当多的不符合要求的方阵后被难住了。他认为任何一个确切解都会引出许多看似不同的解，他不相信问题的所有解都能逃出他的法眼。据此，他说："毫不犹豫，我可以宣称没有人能给出一个完备的36格方阵，同样的结论对于n=10，n=14…也是一样的。一般来说，对于所有奇偶数（oddly even number）⊖问题都是无解的。"

按军团和军衔，欧拉将36名军官命名为$a\alpha$，$a\beta$，$a\gamma$，$a\delta$，$a\varepsilon$，$a\zeta$，$b\alpha$，$b\beta$，$b\gamma$，$b\delta$，$b\varepsilon$，$b\zeta$，$c\alpha$，$c\beta$，$c\gamma$，$c\delta$，$c\varepsilon$，$c\zeta$，$d\alpha$，$d\beta$，$d\gamma$，$d\delta$，$d\varepsilon$，$d\zeta$，$e\alpha$，$e\beta$，$e\gamma$，$e\delta$，$e\varepsilon$，$e\zeta$，$f\alpha$，$f\beta$，$f\gamma$，$f\delta$，$f\varepsilon$，$f\zeta$。他观察到任何一个解都意味着存在两个方阵，一个由拉丁字母组成，另一个由希腊字母组成。每个方阵的行和列上的字母都应不同，由此欧拉开始研究他称之为拉丁方阵的$\{a, b, c, d, e, f\}$的不同构形。仅当一个拉丁方阵和一个希腊方阵是正交的，即当这两个方阵叠加时，（拉丁，希腊）字母对的出现不会多于一次，则可将两个方阵配对成"希腊-拉丁"方阵。例如，若令a=A，b=K，c=Q，d=J，α=♣，β=♠，γ=◇，δ=♡，则与图1等价的拉丁方阵、希腊方阵和希腊-拉丁方阵分别为：

$$\begin{pmatrix} d & a & b & c \\ c & b & a & d \\ a & d & c & b \\ b & c & d & a \end{pmatrix}, \quad \begin{pmatrix} \gamma & \delta & \beta & \alpha \\ \beta & \alpha & \gamma & \delta \\ \alpha & \beta & \delta & \gamma \\ \delta & \gamma & \alpha & \beta \end{pmatrix} \text{和} \begin{pmatrix} d\gamma & a\delta & b\beta & c\alpha \\ c\beta & b\alpha & a\gamma & d\delta \\ a\alpha & d\beta & c\delta & b\gamma \\ b\delta & c\gamma & d\alpha & a\beta \end{pmatrix} \tag{5}$$

当然，对于$n \times n$拉丁方阵，你可以使用任意不同的n个符号，只要任意行和任意列中没有符号出现两次即可。因此也可以使用数值$\{0, 1, \cdots, n-1\}$作为符号。进一步，由于正交是一个对称关系，因此就不需要将方阵划分为拉丁方阵和希腊方阵，由此下文仅使用"拉丁方阵"（以小写字母l表示）。

欧拉作出的关于两个6×6拉丁方阵不可能是正交的断言由Thomas Clausen证实。在1842年8月10日H. C. Schumacher写给高斯的信中提到，Clausen将问题归约为17种不同的情景后得出答案。但Clausen没有发表他的分析结果。最先公开印刷出版论文的是G. Tarry [《Comptes rendus, Association française pour l'avancement des sciences》29，part 2（1901），170—203]，他独立发现了6×6拉丁方阵能划分为17种不同的类。（7.2.3节将研究如何将一个问题分解为组合意义上不等价的类。）

剩下的n=10，n=14等条件的欧拉猜想被人分别"证明"了3次，他们分别是J. Petersen [《Annuaire des mathématiciens》（Paris:1902），413—427]、P. Wernicke [《Jahresbericht der Deutschen Math.-Vereinigung》19（1910），264—267]和H. F. MacNeish [《Annal of Math.》23（1922），221—227]。然而，3个不同的论证都存在着缺陷。甚至在计算机出现多年之后，这个问题还没有得到完全的解决。可以说首个借助机器来求解的组合问题就是搜索10×10的希腊-拉丁方阵：它们存在吗？

在1957年，L. J. Paige和C. B. Tompkins对SWAC计算机进行编程以搜索欧拉

⊖ 所有满足n mod 4 =2的n。——译者注

猜想的反例。他们先“几乎随机”地选择一个特定的10×10拉丁方阵，接着通过程序来寻找其“正交”体。但结果令人沮丧，根据程序先期产生的数据，他们预测完成计算至少需要4.8×10^{11}个小时！因此他们决定5小时之后就关闭计算机。

不久，这个难题被3位数学家——R. C. Bose、S. S. Shrikhande和E. T. Parker突破了，他们找到了一种不同寻常的产生$n \times n$（$n>6$）正交拉丁方阵的方法，解出的拉丁方阵出现在世界主要的报纸上[《Proc.Nat.Acad.Sci.》**45**（1959），734—737，859—862；《Canadian J. Math.》**12**（1960），189—203]。在持续攻关了180多年之后，欧拉的猜想终于被证实是完全错误的。

他们的发现和计算机的帮助完全无关。但Parker在为UNIVAC工作，他将自己的程序设计技巧用于Paige和Tompkins的解决方案，在UNIVAC 1206军用计算机上花费了不到一个小时的时间就得到了问题的答案[参见《Proc. Symp. Applied Math.》**10**（1960），71—83；**15**（1963），73—81]。

让我们仔细看一下早期的程序设计者是如何做的，Parker是如何戏剧性地给出解决方法的。Paige和Tompkins以下面的10×10拉丁方阵$\boldsymbol{L}$以及未知的正交配偶（orthogonal mate）方阵$\boldsymbol{M}$开始。

$$\boldsymbol{L}=\begin{pmatrix}0&1&2&3&4&5&6&7&8&9\\1&8&3&2&5&4&7&6&9&0\\2&9&5&6&3&0&8&4&7&1\\3&7&0&9&8&6&1&5&2&4\\4&6&7&5&2&9&0&8&1&3\\5&0&9&4&7&8&3&1&6&2\\6&5&4&7&1&3&2&9&0&8\\7&4&1&8&0&2&9&3&5&6\\8&3&6&0&9&1&5&2&4&7\\9&2&8&1&6&7&4&0&3&5\end{pmatrix} \text{和 } \boldsymbol{M}=\begin{pmatrix}0&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\1&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\2&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\3&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\4&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\5&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\6&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\7&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\8&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\\9&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup\end{pmatrix} \quad (6)$$

如图所示，不失一般性，我们假定$\boldsymbol{M}$的行以0，1，…，9开始。问题是如何填充剩下的90个空格呢？原先的SWAC程序是从顶到底、从左到右进行计算的。因为0已经出现在$\boldsymbol{M}$的顶行了，左上角的空格不能被填为0；另外，因为对（1，1）已经出现在（$\boldsymbol{L}$，$\boldsymbol{M}$）的第2行的左边，该位置也不能填为1。因此，可以尝试着填为2。接下来的空格，可以填为1。很快地，你就会发现，对于顶行，就字典序而言，满足条件的最小排列是0214365897。类似地，可得下面2行的最小排列，它们分别是1023456789和2108537946；而它们之下的最小合法排列是3540619278。但遗憾的是，接下来就棘手了，接下来的一行由于不存在和先前选择没有冲突的方案而无法列出。因此接下来要将3540619278改为3540629178（但显然这也不行），接着改为3540698172，如此继续，直到改为3546109278为止，而下面的行在失败发生之前可为4397028651。

7.2.3节将研究对这种搜索行为无需实际执行而予以评估的方法。对于这个例子，评估会告诉我们，Paige和Tompkins的方法本质上是对一棵大约包含2.5×10^{18}

个结点的隐式搜索树进行遍历。这些结点中的大多数仅属于树的少数几层，大约有一半处理的是$\boldsymbol{M}$的第6行右边的一半空位，而90个空位中大约有50个是尝试性填写的。为了检查有效性，对搜索树一个典型结点的处理大约需要75次mems（内存存取）。因此在现代计算机上总的运行时间大约是2×10^{20}次mems。

另一方面，Parker使用了欧拉在1779年用于搜索正交配偶的原始方法。他先找出所有被称为$\boldsymbol{L}$的横截（transversal）的东西，即考虑每行取一个元素、每列取一个元素且每一个值只取一次的所有模式。例如，以欧拉的记法，0859734216是一个横截，表示第0列取0、第1列取8，…，第9列取6。每个包含$\boldsymbol{L}$最左列中k的横截代表了将10个k放进方阵$\boldsymbol{M}$的一种正确方式。事实上，找出所有横截的任务是相当容易的，对于上面给定的$\boldsymbol{L}$，恰好有808个横截，对应于k=(0, 1, …, 9)的数目分别是(79, 96, 76, 87, 70, 84, 83, 75, 95, 63)。

一旦知道了横截，剩下的就是10个阶段的与精确覆盖问题相关的工作了，这比原先的(6)中90个阶段的问题要简单得多。所要做的就是用10个不相交的横截覆盖方阵，因为每10个如此的横截集合都等价于一个和$\boldsymbol{L}$正交的拉丁方阵$\boldsymbol{M}$。

事实上，(6)中的方阵$\boldsymbol{L}$恰好有一个正交配偶。

$$\begin{pmatrix}0&1&2&3&4&5&6&7&8&9\\1&8&3&2&5&4&7&6&9&0\\2&9&5&6&3&0&8&4&7&1\\3&7&0&9&8&6&1&5&2&4\\4&6&7&5&2&9&0&8&1&3\\5&0&9&4&7&8&3&1&6&2\\6&5&4&7&1&3&2&9&0&8\\7&4&1&8&0&2&9&3&5&6\\8&3&6&0&9&1&5&2&4&7\\9&2&8&1&6&7&4&0&3&5\end{pmatrix}\perp\begin{pmatrix}0&2&8&5&9&4&7&3&6&1\\1&7&4&9&3&6&5&0&2&8\\2&5&6&4&8&7&0&1&9&3\\3&6&9&0&4&5&8&2&1&7\\4&8&1&7&5&3&6&9&0&2\\5&1&7&8&0&2&9&4&3&6\\6&9&0&2&7&1&3&8&4&5\\7&3&5&1&2&0&4&6&8&9\\8&0&2&3&6&9&1&7&5&4\\9&4&3&6&1&8&2&5&7&0\end{pmatrix}\qquad(7)$$

使用跳舞链接算法，给定808个横截，仅需要1.7×10^{8}次mems计算，就可以证明上述正交配偶是唯一的。进一步，横截查找阶段的耗时大概是5百万mems，相比而言可以忽略。因此，原先的运行时间是2×10^{20} mems，曾被认为是求解一个填写每个空位置都有10^{90}种不同方式的问题所不可避免的，在这里被约简了10^{12}个数量级！。

后面我们还会看到求解如(6)这样的90层问题的进展。事实上，(6)可以直接表示为一个精确覆盖问题（参见习题17），使用7.2.2.1节的跳舞链接技术在花费了1.3×10^{11} mems后就可得到答案。虽然如此，Euler-Parker方法仍比Paige-Tompkins方法快上大约千倍。通过将问题分解为2个相互分离的阶段——横截查找和横截组合，Euler-Parker方法本质上是将计算量从T_1T_2约简为T_1+T_2。

这个故事的警示是：组合问题有着非常庞大的可能性范围，不要过早就予以放弃。一个好的思路会使计算量降低好几个数量级。

谜题和真实世界 本章要研究的许多组合问题，如朗福德对和Ozanam的16张扑克牌问题，最早都是以“脑筋急转弯”式的令人惊异的谜题出现的。许多读者可能会为其所强调的娱乐性而分心，会认为其是琐屑无聊且浪费时间的把戏。读者可能会问，计算机就不能做些有用的工作吗？计算机教科书就不能关注对业界或人类进步有意义的应用吗？

好吧，你正在阅读的这本书的作者是绝对不会反对任何有用的工作或者人类进步这种事的。但他坚信如本书一样的著作应该强调问题的求解方法以及对求解不同问题有所裨益的数学思想和模型，而不应去关注这些方法和模型是否有用、为何有用。我们将学到许多美丽而强大的组合问题求解方法，这些方法的优雅就是我们学习它们的动机。组合挑战随处可见，每天都会出现本章所讨论方法的新的应用。因此不要预先就考虑新的思想对何有用，这会使我们的视野变得局限。

例如，业已证实正交拉丁方阵非常有用，尤其在实验设计方面。早在1788年，François Crettéde Palluel就使用了一个4×4的拉丁方阵来研究羊的喂养，他选取了16只羊，每4只羊属于一个品种，喂以收割于4个不同时期的4种不同食物[《Mémoires d’Agriculture》(Paris: Société Royale d’ Agriculture，trimestre d’été,1788)，17—23]。使用拉丁方阵，他得以使用16只而非64只羊。如果使用希腊-拉丁方阵，他还可以变动另一个参数，比如说，尝试4种不同量的食物或4种不同的放牧方式等。

但如果将讨论聚焦于农牧业，我们可能会涉及饲养细节，陷入蔬菜、谷物以及种植它们的代价等琐事之中。尽管拉丁方阵的应用领域非常广泛（想一下要针对处于5个不同阶段的患有某种疾病的病人，5种不同的体重范围，5个不同的年龄段进行5种不同的药物测试），不是农业工作者的读者可能会决定略过整个主题。而且，完全关注实验设计也可能使读者漏掉拉丁方阵在编码和密码学方面的重要应用（参见习题18—24）。

朗福德对问题，第一眼看去似乎纯粹是个智力游戏，甚至也有其现实应用。T. Skolem曾使用朗福德序列于Steiner元组系统的构造，6.5节已应用该系统于数据库查询[参见《Math. Scandinavica》6（1958），273—280]；20世纪60年代，摩托罗拉公司的E. J. Groth也将朗福德对应用于乘法电路的设计。进一步，能有效发现朗福德对和拉丁方阵横截的算法，如跳舞链接方法等，一般也可应用于精确覆盖问题，而精确覆盖问题和选区选举人公平分摊等重要问题有着紧密的联系。

应用不是最重要之事，谜题也不是。我们的主要目标是将一些基本概念，如拉丁方阵和精确覆盖等，烙进我们的脑海，作为解决明天问题所需要的结构单元、词汇和见识。

然而，讨论问题求解而不解决任何问题是愚蠢的。我们需要好的问题，以或多或少有组织的方式来刺激我们的创造力，激发我们的智力，使我们熟悉基本概念。费神的谜题通常是达到此目标的理想之物，因为几句话就可以描述，无需复杂的背景知识。

Václav Havel曾经说过生命的复杂性是浩瀚无边的，他说：“有太多的东西需

要知道……我们必须抛弃，世界不外乎是一个待解的谜题，一台可使用待发现的指令予以操作的机器或一个可被输入计算机的信息体。”他呼吁人类要增强正义和责任感，提高鉴赏力，具有勇气和怜悯之心。他的言语充满了智慧。感谢上天，我们有谜题可以求解。谜题给我们的生活带来乐趣，和所有其他乐事一样，我们可以适度地沉浸其中。

当然，朗福德和Ozanam是将谜题直接带给了人类而非计算机。但如果我们仅仅是将此类问题交给机器，然后经暴力搜索而非理性思考予以求解，是否少了点什么？George Brewster在1963年写给Martin Gardner的信中，表达了大家所广泛持有的观点，他写道：“将一个用于消遣的谜题输入计算机无异于在小溪中使用炸药来捕获鳟鱼，如此消遣是只有瞬间没有过程的。”

然而，这种观点也遗漏了另外一个重要之处：简单谜题的推广常常超越了人类的能力而唤起我们的好奇心。对推广后的问题的研究也常常揭示了一些可以使用于许多其他问题的有益方法并带来意想不到的效果。的确，这里所研究的许多关键技术源于人们求解各类不同谜题的智力活动。在撰写本章时，作者忍不住谈一下现状，目前的谜题比任何时候都有趣，原因是计算机的速度越来越快，而我们可一直持有威力更猛的炸药。[进一步的评论可参阅作者写于1976年的随笔“游戏问题有用吗?”，参见《Selected Papers on Computer Science》(1996)，169—183]。

谜题确实有问题，它们过于简练。好的谜题应该是数学清晰、构造良好的，但我们还需要学会系统地处理整天都围绕着我们的那些麻烦的、混沌的、有机的东西。确实，一些计算技术之所以重要是因为它们是对付这些复杂性的强有力工具。例如，第5章就给出了图书馆卡按字母进行顺序排列的神秘规则之谜，为了说明仿真技术，2.2.5节以较长的篇幅分析了一个实际电梯控制系统的工作原理。

SGB是作者收集的程序集和数据集，在其之上组合算法所进行的实验可推广于解决众多真实世界中的不同案例。例如，SGB包含了美国的高速公路数据，包括了美国经济的输入输出模型，记录了荷马《伊利亚特》、托尔斯泰《安娜·卡列尼娜》等数部小说的角色表，封装了1879年版的Roget《分类词汇编》，归档了数百所学校的足球比分记录，给出了达·芬奇名画《蒙娜丽莎》的像素灰度值。最重要的是，SGB包含有一个5个字母组成的单词表，详细细节在下面讨论。

5个字母组成的英语单词表 本章许多例子都使用下面这一包含5个字母的单词表

`aargh, abaca, abaci, aback, abaft, abase,`
`abash, ···, zooms, zowie` (8)

(共有5757个单词，数目实在太多，无法在正文中完全列出；但读者可以很容易想象出缺失的部分)。这是作者个人在1972—1992年间收集的，起始原因是作者认为这样的单词表是测试不同组合算法的十分理想的数据集。

单词表只限于收集那些属于英语的词汇是作者有意而为的，因为作者在现实生活中遇到过这些单词。完整的字典则包含着成千个偏僻深奥的词汇，如`aalii`，`abamp`，…，`zymin`和`zyxst`等，这些单词对拼字游戏玩家来说是有用的，但对

不认识这些单词的人来说，它们的出现可能会影响阅读心情。因此，20余年来，作者系统地记录了那些对于《计算机程序设计艺术》这本书的写作似乎是有用的单词。

最后，为了使实验可重复，有必要冻结词汇集。英语语言一直在演化，但5757个SGB单词应是稳定不变的，虽然作者也曾动心，想加入1992年后他才知晓的一些单词，如chads，stent，blogs，ditzy，phish，bling，可能还有tetch等。但是，最终答案是：不，决不。SGB的任何变化都已经结束了，或者说被终结了。

下面的词汇表包含了所有众所周知的英语单词
……它们可用于一个善良的社会，作为人们间联系的纽带。
……要纳入美籍西班牙人的词汇，必须扩充词汇表。
——刘易斯·卡罗尔（Lewis Carroll），《Doublets：A Word-Puzzle》（1879）

如果非要用一个动词来描述“神经兮兮”，看看利利怀特先生你就明白了。
——罗伯特·巴纳德（Robet Barnard），《Corpse in a Gilded Cage》（1984）

SGB不认为如Knuth般的人名是合法单词，但认为gauss（高斯）和hardy是合法的，因为gauss是磁通量单位，而hardy则表示硬汉（hardy）。事实上，SGB完全由通常的英文小写字符构成的单词所组成，不包括含连字符的单词，不包括缩略词以及诸如blasé这种包含音调的单词。因此每个单词可以看成是一个有着5个分量的向量，分量的取值范围为[0..26)，词yucca和abuzz的距离最远，它们间的欧几里得距离是：

$$\|(24,20,2,2,0)-(0,1,20,25,25)\|_2=\sqrt{24^2+19^2+18^2+23^2+25^2}=\sqrt{2415}$$

整个斯坦福图库，包括所有的程序集和数据集，都可从作者的网站（参见封底）下载。SGB中所有单词的列表也很容易得到，其包含在同一网站中的文件sgb-words.txt中。文件有5757行，每个单词一行，以which开始，以pupal结束。单词的默认顺序是单词的使用频率，秩为1000、2000、3000、4000和5000的单词分别为ditch、galls、visas、faker和pismo。依这种秩的定义，本章将使用WORDS(n)表示第n个最常见的词。

偶尔，5个字母的单词表也包括了一些4个字母单词的复数形式。要注意的是，这里不进行维多利亚式的审查。潜在的禁语词汇虽已从官方的拼字玩家字典（《The Official Scrabble® Players Dictionary》）中删除，但不会从SGB中剔除。欲保证含义不雅之词不会出现在专业论文的一种方法是将使用的词汇表限制于WORDS(n)之内，这里的n可取值3000。

习题26－37可用作使用SGB单词表的热身。纵览本章，我们可以看到许多不同的组合算法都使用SGB单词表。例如，对于仍浮现在我们脑中的覆盖问题，4个单词third flock began jumps覆盖了字母表中前21个字母中的20个。5个单词，如{becks，fjord，glitz，nymph，squaw}，最多可覆盖24个不同的字

母，除非使用罕见的非SGB单词，如waqfs（伊斯兰捐助），它和{gyved，bronx，chimp，klutz}一起才可覆盖25个字母。

使用取自WORDS(400)的简单单词就足以构造出一个单词方阵。

```
class
light
agree        (9)
sheep
steps
```

然而，要得到一个单词立方体，选取的单词则几乎要取自WORDS(3000)了。

```
types yeast pasta ester start
yeast earth armor stove three
pasta armor smoke token arena        (10)
ester stove token event rents
start three arena rents tease
```

其中，每个5×5的“切片”都是一个单词方阵。使用简单的跳舞链接算法（参见7.2.2.2节），对于WORDS(3000)，执行了大约3千9百亿次mems之后，可知仅有3个对称的单词立方体，其中之一就像(10)中列出的，其他两个可参阅习题36。令人吃惊的是，对于全部单词的集合，即WORDS(5757)，有83 576个对称立方体。

由单词构成图 将对象排列成行、列、方阵、方体或其他图样既有意思也很重要；但在实际应用中，另一种组合结构则更有意思也更重要，也就是图。回忆一下2.3.4.1节中的描述，一个图是由称为顶点的点以及连接顶点对的称为边的线所组成的。图是普遍存在的对象，已经发现了许多优美的图算法，图自然就成了本章许多小节的主要关注点。事实上，就像其名字所隐含的那样，斯坦福图库基本上就是关于图的程序集和数据集，而收集SGB单词的主要原因也是这些单词可被用于定义有意义且有启发性的图。

路易斯·卡罗尔（Lewis Carroll）在1877年发明了一个称为连字（Word-Links）也称同源异形词（Doublet）的游戏，为基于SGB进行图的研究开辟了道路[参见Martin Gardner，《The Universe in a Handkerchief》（1996），第6章]。之后变得非常流行的Carroll的思路是通过每次修改一个字母的方式，从一个单词得到另一个单词，如：

```
tears—sears—stars—stare—stale—stile—smile        (11)
```

长度最短的转换对应图中的最短路径，其中图的顶点是英语单词，边将“海明距离为1”（意味着它们仅有一个位置不同）的单词联结在一起。

当将使用的单词限制于SGB时，按卡罗尔规则生成的是斯坦福图库中的一个图，斯坦福图库的官方名字为*words*(5757, 0, 0, 0)。每一个由SGB定义的图，都有一个独一无二的标识，称为图的*id*。以卡罗尔规则从SGB导出的图的*id*形为*words*(*n*, *l*, *t*, *s*)，其中*n*是顶点的数目；*l*为0或是一个权重表，用于强调涉及的词汇

种类；t是一个阈值，用于禁止低权重的单词；s是一个伪随机数种子，用于打破由相同权重单词构成的平局。虽然不必要关心所有细节，但下面的例子将给一般性思想。

- *words*(n, 0, 0, 0)表示的图是当卡罗尔规则应用于WORDS(n)时，对于$1 \leqslant n \leqslant 5757$的精确结果。
- *words*(1000, {0, 0, 0, 0, 0, 0, 0, 0, 0}0, s)包含了1000个选自SGB的随机单词，它通常随着s值的不同而不同。
- *words*(766, {0, 0, 0, 0, 0, 0, 0, 1, 0}, 1, 0)包含了在本书作者所著的关于T_EX和METAFONT书中所出现的所有包含5个字母的单词。

在METAFONT图中仅包含766个单词，因此无法构造多个如(11)般长的路径，但

basic—basis—bases—based
—baked—naked—named—names—games (12)

是一个值得记住的例子。

当然，当顶点表示5个字母的单词时，还有许多其他的定义图中边的方法。例如，我们可以要求欧几里得距离小而非海明距离小。或当两个单词共有长度为4的子单词时认为它们是相邻的，该策略实质上丰富了图的构造，甚至使用只和$\mathrm{T_EX}$相关的766个单词也可从chaos生成peace。

chaos—chose—whose—whole—holes—hopes—copes—scope
—score—store—stare—spare—space—paces—peace (13)

（按照这个规则，我们先移去一个字母，然后，可能在不同的位置上，插入另一个字母。）或者我们可以采用一个完全不同的策略，如在词向量$a_1a_2a_3a_4a_5$和$b_1b_2b_3b_4b_5$之间放置一条边，当且仅当点积$a_1b_1+a_2b_2+a_3b_3+a_4b_4+a_5b_5$是某个参数$m$的倍数。基于不同类型数据的图算法可谓繁荣茂盛。

当多重集$\{a_2, a_3, a_4, a_5\} \subseteq \{b_1, b_2, b_3, b_4, b_5\}$，则记$a_1a_2a_3a_4a_5 \rightarrow b_1b_2b_3b_4b_5$（移去第一个字母，插入另一个，再重新排列），那么SGB词汇表也会产生的一类有趣的有向图对象。例如，基于上述规则，我们可以经由长度为6的最短有向路径将words转换为graph。

words→dross→soars→orcas→crash→sharp→graph (14)

理论是实践泰勒级数的首项。
——Thomas M. Cover（1992）

目前图论中所使用的术语体系的数目，
在很大程度上和图论研究者的数目相等。
——Richard P. Stanley（1986）

图论：基本知识 一个图G由一个顶点集V和一个边集E所组成，E中每条边关

联V中两个不同的顶点。除非特别说明，本书假定V和E是有穷集。如果u和v是结点，其满足$\{u, v\}\in E$，则记成$u — v$；而如果u和v是结点，且$\{u, v\}\notin E$，则记成$u \not— v$。满足$u—v$的结点对，称为是“邻居”，即在G中它们是相邻的。由定义得到的一个结论是，$u—v$当且仅当$v—u$。另一个结论是对所有$v\in V$，$v \not— v$，即没有一个顶点和自己相邻。（下面将讨论多重图，它允许顶点到其自身的循环。）

若$V'\subseteq V$且$E'\subseteq E$，则称图$G'=(V', E')$是图$G=(V, E)$的子图。事实上，如果$V'=V'$，G'是G的生成子图。对于给定的顶点子集V'，若E'包含了尽可能多的边，则G'是图G的导出子图。换句话说，当$V'\subseteq V$的，由V'所导出的$G=(V, E)$的子图是$G'=(V', E')$，其中

$$E'=\{\{u, v\} \mid u\in V', v'\in V', \text{且}\{u, v\}\in E\} \tag{15}$$

子图G'记为$G|V'$，通常称为“限制于V'的G”。一般地，当$V'=V\setminus\{v\}$时，$G|(V\setminus\{v\})$可简记为$G\setminus v$（“从G中去除顶点v”）。类似地，记法$G\setminus e$当$e\in E$时，用于表示子图$G'=(V, E\setminus\{e\})$，它是从G中移去一条边而非一个顶点所得的图。注意，所有前面描述过的称为$words(n, l, t, s)$的SGB图，都是主图$words(5757, 0, 0, 0)$的导出子图，这些图是仅仅改变词汇而非邻接规则的结果。

称一个有n个顶点、e条边的图是阶（order）为n、规模（size）为e的图。最简单且最重要的阶为n的图是完全图K_n、路径P_n和回路C_n。假定图的顶点集为$V=\{1, 2, \cdots, n\}$，则

- K_n有$\binom{n}{2}=\frac{1}{2}n(n-1)$条边$u—v$，对于$1\leqslant u<v\leqslant n$；每一个$n$顶点的图都是$K_n$的生成子图（spanning subgraph）。
- P_n有$n-1$条边$v—(v+1)$，对于$1\leqslant v<n$，当$n\geqslant 1$时；P_n是一条从1到n的长度为$n-1$的路径。
- C_n有n条边$v—((v \bmod n)+1)$，对于$1\leqslant v\leqslant n$；仅当$n=0$或$n\geqslant 3$时，C_n是图（但C_1和C_2是多重图）。

实际上，我们已经在顶点集$\{0, 1, \cdots, n-1\}$或在非$\{1, 2, \cdots, n\}$的任意n个元素的V集合上定义了K_n、P_n和C_n，因为仅仅顶点名字不同、而边的结构相同的两个图是组合等价的。

从形式上讲，如果存在从V到V'的一个一一对应的映射φ，满足G中有边$u—v$当且仅当G'中有边$\varphi(u)—\varphi(v)$，则称图$G=(V, E)$和$G'=(V', E')$是同构的（isomorphic）。通常使用记法$G\cong G'$表示G和G'是同构的。本书就不如此严格了，我们将同构的图看成是相同的，尽管G和G'的顶点集不是严格相等的，我们也写成$G=G'$。

简单的图可以通过画出图形予以定义，在图形中顶点表示为小圆圈，边表示为其间的线条。图2所示的是几个重要的例子，它们的性质将在后面讨论。图2e中的Petersen图是以Julius Petersen的名字命名的，Julius Petersen是一位早期的图论学者，他使用图作为工具对一个似是而非的猜想进行反驳[《L'Intermédiaire des Mathématiciens》5（1898），225—227]，事实上，Petersen图是许多关于一般图的性质的乐观猜测的重要反例。图2f中的Chvátal图，是由Václav Chvátal在《J. Combinatorial Theory》9（1970），93—94一文中提出的。

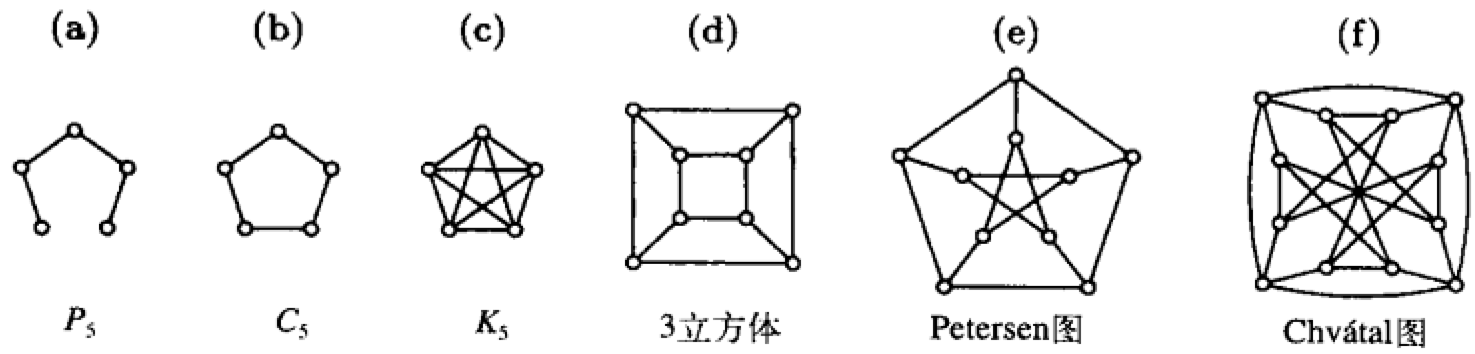
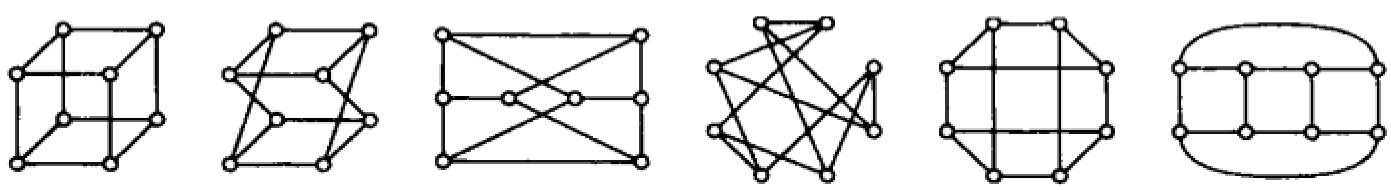


图2　6个图例，分别有（5,5,5,8,10,12）个顶点和（4,5,10,12,15,24）条边

图形中的线条交接于非顶点的点是允许的。例如，图2f中的中心点并非Chvátal图的顶点。如果存在一种使线条不相交的图形画法，则对应的图称为平面图。显然，P_n和C_n总是平面图；图2d所示的3立方体也是平面图。但K_5由于有太多边，因此不是平面图（参见习题46）。

一个顶点的度是与其邻接的顶点数。如果所有的顶点的度都是一样的，则称该图是正则的（regular）。在图2中，由于有2个顶点的度为1，有3个顶点的度为2，因此，P_5是非正则的，而剩下的5个图都是正则的，分别有度(2,4,3,3,4)。一个度为3的正则图通常称为“立方体”或“三价体”。

对于给定的图，有多种绘制方式，其中一些绘制方式比其余的更明白易懂。例如，下面6个图：

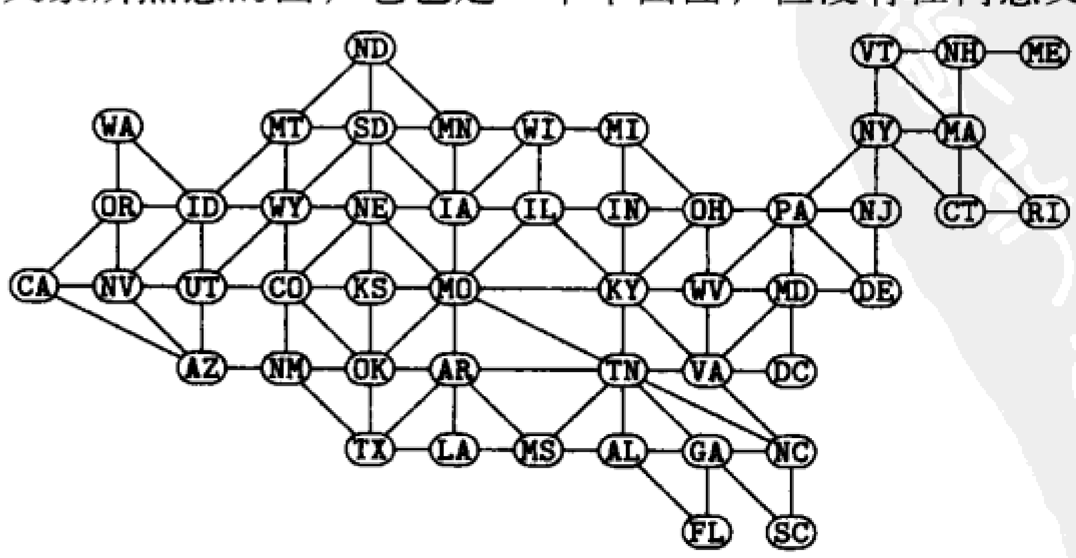


(16)

都和图2d所示的3立方体是同构的。图2f中Chvátal图的布局是Chvátal的论文发表多年后由Adrian Bondy给出的，该布局揭示了若干未曾预期的对称性。

图的对称性也被称作自同构（automorphism），是保持邻接的顶点的一个排列，换句话说，若G中有边u—v时也有边$\varphi(u)$—$\varphi(v)$，则排序φ是G的一个自同构。如图2f般精心绘制的图形可以揭示出潜在于图中的对称性，但一个单一的图并非总是可以展示存在于其中的对称性。例如，3立方体有48个自同构，而Petersen图有120个自同构。我们将在7.2.3节讨论同构和自同构。开发对称性的用处是可避免不必要的重复计算，对于一个有k个自同构的图，利用对称性可使算法几乎快上k倍。

现实世界所形成的图和图2所示的纯粹数学意义上的图有着很大的不同。例如，下面就是一个大家所熟悉的图，它也是一个平面图，但没有任何意义上的对称性。



(17)

该图表示的是美国各州相连的情景，后面很多例子还会用到它。为了方便，图中的49个顶点用邮政编码而非简略的空圈予以标记。

路径和回路 称一个图的生成路径为哈密顿路径（Hamiltonian path），称一个图的生成回路为哈密顿回路（Hamiltonian cycle）以纪念W. R. Hamilton，Hamilton在1859年发明并开始销售一种12面体智力玩具，游戏的目标就是在12面体的边上寻找此种路径和回路。T. P. Kirkman在《Philosophical Transactions》148（1858），145—161（参阅由N. L. Biggs、E. K. Lloyd和R. J. Wilson（1998）所写的《Graph Theory 1736—1936》，第2章）对一般多面体上的此类问题进行了独立的研究。实际上，寻找生成树和回路的问题是一个古老的问题，我们有理由认为它是最古老的组合学问题之一，因为国际象棋中马的移动路径和棋盘遍历问题在印度历史中可连续追溯到9世纪（参见7.3.3节）。称一个包含哈密顿回路的图为哈密顿图（Hamiltonian）。（巧合的是，Petersen图是一个既非平面图也非哈密顿图的最小3正则图；参阅C. de Polignac，《Bull. Soc. Math. de France》27（1899），142—145。）

一个图的周长（girth）是图所包含的最短回路的长度；如果一个图是非循环的（即不含回路），则其周长是无穷的。例如，图2中6个图的周长分别是（∞, 5, 3, 4, 5, 4）。不难证明最小度为5、周长为5的图至少有k^2+1个顶点。进一步的分析表明，事实上该最小值只在$k=2$（C_5）、$k=3$（Petersen）、$k=7$或$k=57$时取得（参见习题63和65）。

两个顶点u和v的距离（distance）$d(u, v)$是图中从u到v的最短路径的长度；如果不存在此种路径，则定义距离为无穷。显然，$d(v, v)=0$且$d(u, v)=d(v, u)$。我们还有下面的三角不等式：

$$d(u, v)+d(v, w)\geqslant d(u, w) \tag{18}$$

如果$d(u, v)=p$且 $d(v, w)=q$，其中$p<\infty$且$q<\infty$，则有路径：

$$u=u_0—u_1—\cdots—u_p=v \text{ 且 } v=v_0—v_1—\cdots—v_q=w \tag{19}$$

而我们可以找到满足$u_r=v_s$的最小下标r，其中s也是一个下标，使

$$u_0—u_1—\cdots—u_{r-1}—v_s—v_{s+1}—\cdots—v_q \tag{20}$$

是一个从u到w的长度小于等于$p+q$的路径。

图的直径（diameter）是所有顶点对u和v间距离$d(u,v)$的最大者。如果直径是有穷的，则图是连通的（connected）。一个图的顶点总是可以划分为连通分量（component），对于两个顶点u和v，当且仅当$d(u, v)<\infty$，它们属于同一连通分量。

例如，在图*words*(5757, 0, 0, 0)中，由于(11)是`tears`到`smile`的最短路径，因此*d*(`tears`, `smile`)=6。同理*d*(`tears`, `happy`)=6，*d*(`smile`, `happy`)=10，*d*(`world`, `court`)=6。但*d*(`world`, `happy`)=∞，即图不是连通的。事实上，该图包含了671个如`aloof`这样的没有邻居的单词，这些单词自身形成了阶为1的连通分量。进一步如`alpha`—`aloha`、`droid`—`druid`和`opium`—`odium`这样的对形成了阶为2的103个连通分量。其中一些阶为3的连通分量，如`chain`—`chair`—

choir是路径，而{getup, letup, setup}是回路。还有一些小的连通分量，如奇异路径

$$\text{login—logic—yogic—yogis—yogas—togas} \tag{21}$$

该路径上的所有顶点均没有其他的邻居。大部分的5字单词均属于一个阶为4493的庞大连通分量，如果从一个给定的单词跨出2步，那么和原先相比，你的单词和庞大连通分量中的单词相连接的可能性就会大上15倍。

类似地，对于图*words*(*n*, 0, 0, 0)，当*n*=(5000, 4000, 3000, 2000, 1000)时分别有阶为(3825, 2986, 2056, 1198, 224)的庞大连通分量。当*n*较小时，就没有足够的用于提供连通性的边了。例如，*words*(500, 0, 0, 0)有327个不同的连通分量，没有一个阶是15或以上的。

可以将距离的概念推广到任意值k，即可定义$d(v_1, v_2, \cdots, v_k)$ 为一个连通子图中包含顶点$\{v_1, v_2, \cdots, v_k\}$的最少边数。例如，由于子图

```
blood—brood—broad—bread—tread—treed—tweed
                                |     |
tears—teams—trams—trims—tries—trees  tweet      (22)
                                      |
                              sweat—sweet
```

有15条边，且没有含14条边的更小子图，因此d(blood, sweat, tears)为15。

在2.3.4.1节中，我们称一个有着最少的边的连通图为自由树（free tree）。一个和推广后的距离$d(v_1, v_2, \cdots, v_k)$相对应的子图必定是自由树。由于Jacob Steiner在欧几里得平面中曾提到k=3时点集$\{v_1, v_2, v_3\}$的情况，因而自由树曾一度被误称为Steiner树（Steiner tree）[《Crelle》13（1835），362—363]。Franz Heinen在《Über Systeme von Kräften》（1834）中给出了问题的解；高斯在1836年3月21日给Schumacher的信中将分析的结果扩充到k=4的情景。

着色 如果一个图可以分解为k个或更少的若干部分，使每条边所关联的顶点属于不同的部分，或等价地说，可最多使用k种不同颜色对图的顶点进行着色，使任何相邻顶点所着的颜色不同，则称图是k分图（k-partite）或是k可着色的（k-colorable）。由F. Guthrie于1852年提出的猜想，最后由K. Appel、W.Haken和J. Koch使用计算机辅助证明了著名四色定理[《Illinois J. Math.》21（1977），429—567]，表明每个平面图都是4可着色的。虽然对该猜想目前还没有一个简单的证明，但如(17)般的特例是可以轻易予以4着色的（参见习题45），并且，对于一个一般的平面图，$O(n^2)$步就足以进行4着色了[N. Robertson、D. P. Sanders、P. Seymour和R. Thomas，《STOC》28（1996），571—575]。

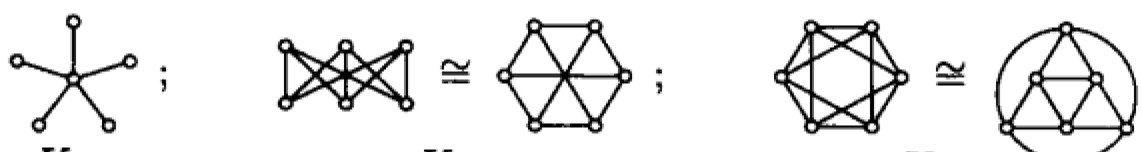
在实际中，图的2着色问题尤其重要。一个2分图一般被称为二分图（bipartite），或简称为“偶图”，偶图中的每部分只含每边的一个顶点。

定理B 一个图是二分图当且仅当其不含奇数长度的回路。

证明：[参阅D. König，《Math. Annalen》77（1916），453—454]显然，k分图的每个子图也是k分图。因此仅当C_n本身是一个二分图时其才有可能是一个二分图的子图，此时n必为偶数。

反之，如果一个图不含奇数长度的回路的话，我们就可以按照如下方式使用两种颜色{0, 1}对其顶点进行着色：开始时令所有顶点都是未着色的。如果已着色顶点的所有邻居顶点都已被着色，则选择一个未着色的顶点，并将其着色为0。否则，选择一个以未着色顶点v为邻居的已着色顶点u，将v着色为与该邻居相反的颜色。习题48将证明该过程结束时我们将得到一个有效的2着色图。▮

完全二分图（complete bipartite graph）$K_{m,n}$是顶点可分为规模为m和n两个部分的最大的二分图。我们可在顶点集$\{1, 2, \cdots, m+n\}$上定义完全二分图，即若$1\leqslant u\leqslant m<v\leqslant m+n$则$u$—$v$。换句话说，$K_{m,n}$有$mn$条边，每条边都是从第一个部分取一个顶点接着从第二个部分取一个顶点的结果。类似地，完全k分图（complete k-partite graph）$K_{n_1,\cdots,n_k}$有$N=n_1+\cdots+n_k$个顶点，划分成规模为$\{n_1,\cdots,n_k\}$的部分，任何不属于同一部分的两个顶点间的连接都是图的边。下面是$N=6$时的一些例子。



$K_{1,5}$; $K_{3,3}$; $K_{2,2,2}$ (23)

注意，$K_{1,n}$是自由树，俗称为阶为$n+1$的星图（star graph）。

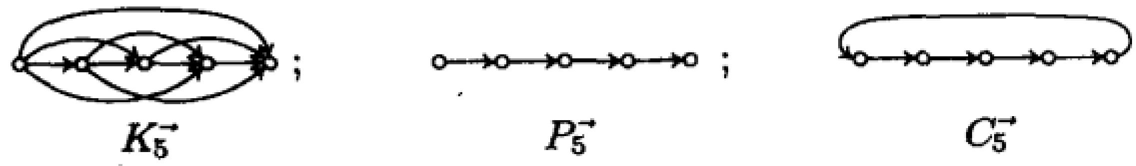
从现在开始，称“含有向边的图”为“有向图”，
这种叫法既清楚又简短，必将流传。
——George Pólya，给Frank Harary的信（c. 1954）

有向图 2.3.4.2节定义了有向图（directed graph或digraph），除了连接顶点的是弧（arc）而非边外，其他和图相似。弧$u\to v$是从一个顶点到另一个顶点的有向边，而边$u\to v$只是将两个顶点连接在一起，并不考虑顶点间的顺序。进一步地，我们允许有向图包括一个顶点到自身的自循环$u\to u$，也允许相同顶点u和v间存在多条弧$u\to v$。

从形式上说，一个阶为n、规模为m的有向图$D=(V, A)$是一个有n个顶点的集合V以及一个含m个有序对(u, v)的多重集A，其中$u\in U$且$v\in V$。有序对称为弧，当$(u, v)\in A$时记$u\to v$。若A是一般集合而非多重集时，即对于所有的u和v，其间最多只有一条弧，则称该有向图是简单的（simple）。与每条弧(u,v)关联的是起始顶点u和终止顶点v，后者也称为弧的“末端”（tip）。每个顶点都有出度（out-degree）$d^+(v)$，即以v为起始顶点的弧的数目，也有入度（in-degree）$d^-(v)$，即以v为末端的弧的数目。入度为0的顶点称为“源”，出度为0的顶点称为“汇”。注意，$\sum_{v\in V}d^+(v)=\sum_{v\in V}d^-(v)$，原因是两者都等于弧的总数$m$。

我们所定义的图的记法大都可以自然地延续到有向图，只需在有必要区分弧和边的情况下插入“有向的”或“有方向的”即可。例如，有向图的有向子图可以是生成的、导出的或是通过其他方式得到的。有向图$D=(V, A)$和$D'=(V', A')$间的同构关系是V到V'的一一对应φ，满足对所有$u,v\in V$，D中弧$u\to v$的数目等于D'中$\varphi(u)\to\varphi(v)$的数目。

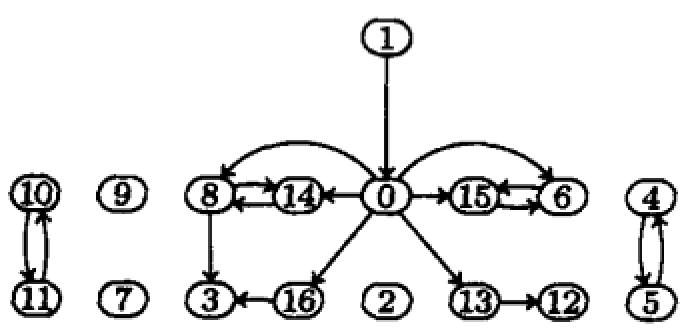
有向图的图形是顶点间带箭头，而非不带任何装饰的线条。最简单而重要的阶为n的有向图是图K_n、P_n和C_n的有向变体，即传递竞赛图（transitive tournament）$\vec{K_n}$、有向路径$\vec{P_n}$和有向回路$\vec{C_n}$。对于n=5，它们的图示如下。



(24)

完全有向图（complete digraph）J_n，是包含n个顶点的最大的简单有向图；对于u和v的每种选择都有着对应的弧，因此共有n^2条弧$u \to v$。

图3所示是一个更复杂的图形，其对应阶为17的有向图，是由阿加莎·克里斯蒂（Agatha Christie）的小说《东方快车谋杀案》《Murder on the Orient Express》（1934）中的人物比利时侦探波罗（Hercule Poirot）描绘的，因此可称为“东方快车图”。图中顶点对应于故事中伊斯坦布尔—加来（Stamboul-Calais）卧铺车厢中的包厢，弧$u \to v$表示包厢u的居住者其不在犯罪现场的证人在包厢v中。该例子有6个连通分量，即{0, 1, 3, 6, 8, 12, 13, 14, 15, 16}、{2}、{4, 5}、{7}、{9}和{10, 11}，这里有向图的连通性是通过将弧看成边而予以判定的。



0: Pierre Michel，法国乘务员
1: Hercule Poirot，比利时侦探波罗
2: Samuel Edward Ratchett，死去的美国人
3: Caroline Martha Hubbard，美国女舍监
4: Edward Henry Masterman，英国男仆
5: Antonio Foscarelli，意大利汽车推销员
6: Hector MacQueen，美国秘书
7: Harvey Harris，没有露面的英国人
8: Hildegarde Schmidt，夫人的德国女仆
9: （空）
10: Greta Ohlsso，瑞典护士
11:Mary Hermione Debenham，英国女家庭教师
12:Helena Maria Andrenyi，美丽的伯爵夫人
13:Rudolph Andrenyi，匈牙利伯爵/外交官
14:Natalia Dragomiroff，俄国公主
15:Colonel Arbuthnot，来自印度的英国军官
16:Cyrus Bettman Hardman，美国侦探

图3 阿加莎·克里斯蒂描绘的阶为17、规模为18的有向图

如果第一条弧的末端是第二条弧的起点，则称两个弧是连续的（consecutive）。弧的连续序列（$a_1, a_2,\cdots, a_k$）称为长度为k的走访（walk），可以将顶点和弧一起予以符号化表示。

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \cdots v_{k-1} \xrightarrow{a_k} v_k \tag{25}$$

对于简单有向图，只要指明顶点就够了，例如1→0→8→14→8→3是图3中的一个走访。当顶点{$v_0, v_1, \cdots, v_k$}不同时，(25)中的走访是一个有向路径；当除v_k=v_0外其余均不同时，它是一个有向回路。

在一个有向图中，有向距离$d(u, v)$是从u到v的最短有向路径中弧的数目，也是从u到v的最短走访长度。$d(u, v)$的值可能和$d(v, u)$不同，但（18）中的三角不等式仍然成立。

由于每条边u—v本质上等价于成对出现的弧$u \to v$和$v \to u$，因此每个图都可以被看成是一个有向图。以此方式得到的有向图仍保留原来图的所有性质，例如，图中每个顶点的度就成了有向图中顶点的出度，也是有向图中顶点的入度。而且距离也是一样的。

多重图（multigraph）(V, E)与图类似，只是边E可以是任意顶点对$\{u, v\}$的多重集；顶点到自身的循环v—v，其对应“多重对”（multipairs）$\{v, v\}$，也是允许的。例如



(26)

是一个阶为3，有6条边$\{1, 1\}$、$\{1, 2\}$、$\{2, 3\}$、$\{2, 3\}$、$\{3, 3\}$和$\{3, 3\}$的多重图。图中顶点的度是$d(1)=d(2)=3$，因为每个循环都给度的计数贡献了2，因此$d(3)=6$。当将多重图看成有向图时，循环边v—v就成了两条循环弧$v \to v$。

图和有向图的表示 任意有向图，进一步说是任意图或多重图，都可由其邻接矩阵（adjacency matrix）$A=(a_{uv})$所完全描述，对应于n个顶点，矩阵有n行n列。每个矩阵元素a_{uv}说明了从u到v的边的数目。例如，$K_3^{\to}$，$P_3^{\to}$，$C_3^{\to}$，J_3和（26）的邻接矩阵分别是

$$K_3^{\to} = \begin{pmatrix}011\\001\\000\end{pmatrix},\ P_3^{\to} = \begin{pmatrix}010\\001\\000\end{pmatrix},\ C_3^{\to} = \begin{pmatrix}010\\001\\100\end{pmatrix},\ J_3 = \begin{pmatrix}111\\111\\111\end{pmatrix},\ A = \begin{pmatrix}210\\102\\024\end{pmatrix} \qquad (27)$$

使用矩阵理论这一强大的数学工具，通过研究邻接矩阵可证明许多非明显的图的性质；习题65列出了一些我们可以完成的特别突出的例子。其中的一个原因是，关于有向图矩阵乘法有一个简单的解释。考虑A的平方，由定义，其第u行第v列的元素是

$$(A^2)_{uv} = \sum_{w \in V} a_{uw} a_{wv} \qquad (28)$$

因为a_{uw}是从u到w的弧的数目，可以看出$a_{uw}a_{wv}$是形为$u \to w \to v$的走访的数目。因此$(A^2)_{uv}$是长度为2的从u到v的总走访数。类似地，A^k的元素是任意顶点对间长度为k的总走访数，这里$k \geqslant 0$。例如，（27）中的矩阵A满足

$$A = \begin{pmatrix}2&1&0\\1&0&2\\0&2&4\end{pmatrix},\quad A^2 = \begin{pmatrix}5&2&2\\2&5&8\\2&8&20\end{pmatrix},\quad A^3 = \begin{pmatrix}12&9&12\\9&18&42\\12&42&96\end{pmatrix} \qquad (29)$$

此即在多重图（26）中有12种长度为3的从顶点1到顶点3的走访，有18种从顶点2到其自身的长度为3的走访。

重排顶点的操作对应着将矩阵A变换为$P^{-}AP$，其中P是一个置换矩阵（一个每行每列都恰好只有一个1的0−1矩阵），而$P^{-}=P^{\mathrm{T}}$是逆置换。因此

$$\begin{pmatrix}210\\102\\024\end{pmatrix},\ \begin{pmatrix}201\\042\\120\end{pmatrix},\ \begin{pmatrix}012\\120\\204\end{pmatrix},\ \begin{pmatrix}021\\240\\102\end{pmatrix},\ \begin{pmatrix}402\\021\\210\end{pmatrix} \text{和} \begin{pmatrix}420\\201\\012\end{pmatrix} \qquad (30)$$

是（26）的所有邻接矩阵，没有别的了。

对于阶n，当$n>1$时，有多于$2^{n(n-1)/2}/n!$个阶为n的图。使用最经济的编码方法，它们之中也几乎全部都需要$\Omega(n^2)$个比特位作为数据存储。从内存使用的观点看，在一台计算机内表示大多数可能出现的图的最好方式是使用邻接矩阵。

但从实际问题中所产生的图和从所有可能的集合中随机选择的图有着非常不同的性质。一个现实世界中的图通常是“稀疏的”，比如说有$O(n\log n)$条边而非$\Omega(n^2)$条边，除非n相当小，$\Omega(n^2)$比特位的数据存储要求是很少发生的。例如，假定人对应于顶点，边对应于友谊，以50亿人作为考虑，很少有人会有10000位以上的朋友。就算每个人平均有10 000位朋友，图仍然只有2.5×10^{13}条边，而几乎所有阶为50亿的图都大约有6.25×10^{18}条边。

因此在一台计算机内部表示一个图的最好方式和记录一个邻接矩阵n^2个a_{uv}值的方式是非常不同的。不过，斯坦福图库上的算法所使用的数据结构，除了某些简化外，和2.2.6节所讨论的稀疏矩阵的链表表示法是类似的。该方法已被证实不仅通用有效而且易于使用。

有向图的SGB表示法结合了顺序分配和链接分配技术，使用了两种基本类型的结点。一些结点用于表示顶点，其他结点则用于表示弧。（对于需要同时处理几个图的算法，需要第3类结点，用于表示全局。但对于每个图，仅仅需要一个图结点，因此顶点和弧对应的结点仍处于支配地位。）

下面描述它的工作流程：每个阶为n、规模为m的SGB图都建立在一个有n个顶点结点的顺序数组之上，由此可容易地对顶点k进行存取，这里$0\leqslant k<n$。与之相反，m个弧结点则相互链接于无结构的一般内存池之中。典型地，每个顶点结点占用32个字节，每条弧占用20个字节（图结点占用220个字节）；这里虽然列出了各种结点占用的具体字节数，但要对结点规模进行修改不存在任何困难。每个结点上有数个域是固定的，它们在所有情况之下含义都是明确的；其他的域则在不同的算法或一个算法的不同阶段可用于不同的目的。称一个结点有着固定用途的部分为“标准域”，有着多重用途的部分为“应用域”（utility field）。

每个顶点结点有2个称为NAME和ARCS的标准域。如果v是一个指向顶点结点的变量，则称v是一个顶点变量（vertex variable）。NAME(v)指向的是一个字符串，在面向人的输出格式下，该字符串是相应顶点的标识；例如（17）中的49个顶点就有着类似于CA，WA，OR，…，RI这样的名字。另外一个标准域ARCS(v)在算法中尤为重要，它指向长为$d^+(v)$的单链表中的第一个弧结点，该单链表中的每个结点都对应着从顶点v传出的一条弧。

每个弧结点有2个分别称为TIP和NEXT的标准域。如果a是一个指向弧结点的变量，则称a是一个弧变量（arc variable）。TIP(a)指向的是一个顶点结点，它是弧a的末端；NEXT(a)则指向一个弧结点，它对应起始顶点为a的下一条弧。

出度为0的顶点可通过令ARCS(v)=Λ（空指针）来表示。否则，比如说出度为3，则数据结构包含3个弧结点，ARCS(v)=a_1，NEXT(a_1)=a_2，NEXT(a_2)=a_3，NEXT(a_3)=Λ；而相应的图中从v出来有3条指向TIP(a_1)、TIP(a_2)和TIP(a_3)的弧。

例如，假定想计算顶点v的度并将其存于名为ODEG的应用域中，我们可如下

进行。

$$
\begin{aligned}
&\text{设} a \leftarrow \texttt{ARCS}(v) \text{且} d \leftarrow 0 \\
&\text{当} a \neq \Lambda \text{时，设} d \leftarrow d+1 \text{且} a \leftarrow \texttt{NEXT}(a) \\
&\text{设} \texttt{ODEG}(v) \leftarrow d
\end{aligned} \tag{31}
$$

当将图或多重图看成有向图时，如上所述，边u—v就等价于两条弧$u \to v$和$v \to u$。这些弧被称为“伴侣”，占用了两个弧结点，比如说是a和a'，其中a出现在从u出发的弧结点列表中，而a'出现在从v出发的弧结点列表中。此时有TIP(a)=v且TIP(a')=u。在算法中，如果想从一个表快速移动到另一个表，代码为

$$\texttt{MATE}(a) = a' \text{ 且 } \texttt{MATE}(a') = a \tag{32}$$

然而，通常我们不想存储从一个弧到其伴侣的显式指针，也不想在每个弧结点中使用一个叫做MATE的应用域，通过对数据结构进行巧妙构建，就可将所需的链接蕴涵于通常的表示法中。

隐式伴侣的工作诀窍为：当创建无向图或多重图的边u—v时，对于$u \to v$和$v \to u$，我们引入连续（consecutive）的弧结点。例如，如果每个弧结点占用20个字节，就为每个新的弧结点对保留40个连续的字节。假定也可保证内存地址的第一个字节是8的倍数，则如果弧结点a在内存的地址是α，则其伴侣的地址是

$$\left.\begin{cases} \alpha + 20, \text{如果 } \alpha \bmod 8 = 0 \\ \alpha - 20, \text{如果 } \alpha \bmod 8 = 4 \end{cases}\right\} = \alpha - 20 + (40 \mathbin{\&} ((\alpha \mathbin{\&} 4) - 1)) \tag{33}$$

对于组合算法，一些操作可能会被执行上万亿次，假定一次操作可省略3.6ns，则计算可快上一个小时，因此这种技巧对于组合算法是很有价值的。但（33）不能从一个实现直接移植到另一个实现，例如，当弧结点的规模从20改为24时，我们不得不将（33）中的40、20、8和4改为48、24、16和8。

本书中的算法不对结点的规模作任何假设，我们采用的是C语言及其后继的一些惯用法，即如果a是指向一个弧结点的指针，则‘a+1’是指向内存中紧随该弧结点之后的弧结点的指针。一般地，如果每个弧结点有c个字节，则

$$\text{LOC}(\text{NODE}(a+k)) = \text{LOC}(\text{NODE}(a)) + kc \tag{34}$$

类似地，如果v是一个顶点变量，则‘$v+k$’指向的是紧随v其后的第k个顶点结点，该结点的内存实际位置是v加上k乘以顶点结点的规模。

图结点g的标准域有M(g)，表示弧的总数；N(g)，表示顶点的总数；VERTICES(g)，指向所有顶点结点的顺序列表中的第一个结点；而ID(g)，是如words(5757, 0, 0, 0)这样的字符串，它是图的标识；还有其他的域，如用于图扩展或收缩时内存分派或回收的域，协助将图输出为外部格式的域，作为图和其他用户或对图进行操作的其他系统间的接口的域。但本书很少会提及这些图结点域。另外，在本章，几乎所有的图算法都是在较高的抽象层次上使用英语描述的，不涉及机器程序所处的比特层，所以这里就没有必要给出SGB的完全描述了。

一个简单的图算法 为了说明将出现于后面章节中的高层次算法的类型，这里将定理B转化为一步步执行的过程。当图是二分图时，该过程使用两种颜色对给定图的顶点进行着色。

算法B（二分性测试，bipartiteness testing） 给定一个表示为SGB格式的图，算法或者对于每个符合COLOR(v)∈{0, 1}的顶点v找出一个2着色方案，或者在没有2着色可能性时以告知着色失败终止。这里COLOR是每个顶点结点所含的一个应用域。另一个应用域LINK(v)是一个顶点指针，用于维护所有已着色的、但邻居还未被检查的顶点的栈。辅助顶点变量s是指向栈顶的指针。算法中还有用于顶点的变量u、v和w，用于弧的变量a。顶点结点假定位于v_0+k中，$0\leqslant k<n$。

B1. [初始化。] 对$0\leqslant k<n$，令COLOR(v_0+k)←-1(现在所有顶点都是未着色的)。接着令$w\leftarrow v_0+n$。

B2. [结束了？] （此时，所有$\geqslant w$的顶点及其邻居都已被着色。）如果$w=v_0$则以成功终止。否则，令$w\leftarrow w-1$，即下一个要处理的顶点结点。

B3. [在必要时对w着色。] 如果COLOR(w)≥0，返回B2；否则令COLOR(w)←0，LINK(w)←Λ，$s\leftarrow w$。

B4. [栈⇒u。] 令$u\leftarrow s$，$s\leftarrow$LINK(s)，$a\leftarrow$ARCS(u)。（我们将检查已着色顶点u的所有邻居顶点。）

B5. [u处理好了吗？] 如果a=Λ，转B8。否则令$v\leftarrow$TIP(a)。

B6. [处理v。] 如果COLOR(v)<0，令COLOR(v)←$1-$COLOR(v)，LINK(v)←s，$s\leftarrow v$；否则，若COLOR(v)=COLOR(u)，则以失败终止。

B7. [对a循环。] 令$a\leftarrow$NEXT(a)并返回B5。

B8. [栈非空？] 如果$s\neq$Λ，返回B4；否则返回B2。▌

该算法是通常被称为“深度优先搜索”的图遍历过程的一个变体，我们将在7.4.1节研究该遍历算法。算法的运行时间是$O(m+n)$，其中m是弧数，n是顶点数（参见习题70），因此该算法很适合于通常的稀疏图。稍加改变，我们就可使其在以失败终止时输出一个奇数长度的循环，并以此佐证给定的图不存在2着色可能性（参见习题72）。

图的例子 斯坦福图库包括了30多个图的生成例程，可生成大量用于实验的无向图和有向图。前面已经讨论了图*words*，为了感受一下其他图的情况，我们来看一些例子。

- *roget*(1022, 0, 0, 0)是一个有1022个顶点和5075条弧的有向图。顶点表示单词或概念所属的范畴，这些单词包括在著名的由P. M. Roget和J. L. Roget所撰写的19世纪同义词词典《Thesaurus》（London: Longmans Green, 1879）中。弧是图中诸范畴间的交叉引用。例如，典型的弧有water→moisture，discovery→truth，preparation→learning，vulgarity→ugliness，wit→amusement。
- *book*（“jean”, 80, 0, 1, 356, 0, 0, 0）是一个有80个顶点和254条边的图。顶点表示维克多·雨果《悲惨世界》中的人物；边连接了小说中曾相互邂逅的人物。典型的边为Fantine—Javert，Cosette—Thénardier。
- *bi_book*（“jean”, 80, 0, 1, 356, 0, 0, 0）是一个有80+356个顶点和727条边的二分图。顶点表示维克多·雨果《悲惨世界》中的人物或章节，边连接了

在小说中的人物和其在小说中出现的章节（例如，Napoleon—2.1.8，Marius—4.14.4）。

• *plane_miles* (128, 0, 0, 0, 1, 0, 0)是一个有着129个顶点和381条边的平面图。顶点表示美国或加拿大的128个城市，以及一个表示无穷远点的特殊顶点INF。边定义了城市间基于平面经纬度的所谓德朗奈三角关系（Delaunay triangulation），这意味着*u*—*v*当且仅当通过*u*和*v*的最小圆圈不包含任何其他的顶点。INF和包含所有城市位置的凸包的所有顶点间也有边。典型的边为Seattle, WA—Vancouver, BC—INF；Toronto, ON—Rochester, NY。

• *plane_lisa* (360, 250, 15, 0, 360, 0, 250, 0, 0, 2295000)是一个有着3027个顶点和5967条边的平面图。该图始于列奥纳多·达·芬奇的著名油画《蒙娜丽莎》的数字化结果，包含了360（行）×250（列）个像素，对每个像素亮度从0（黑）到15（白）共16级进行取整。结果有3027个亮度相等的连贯区域，当两个区域共享一个像素的边界时定义它们是相邻的（参见图4）。

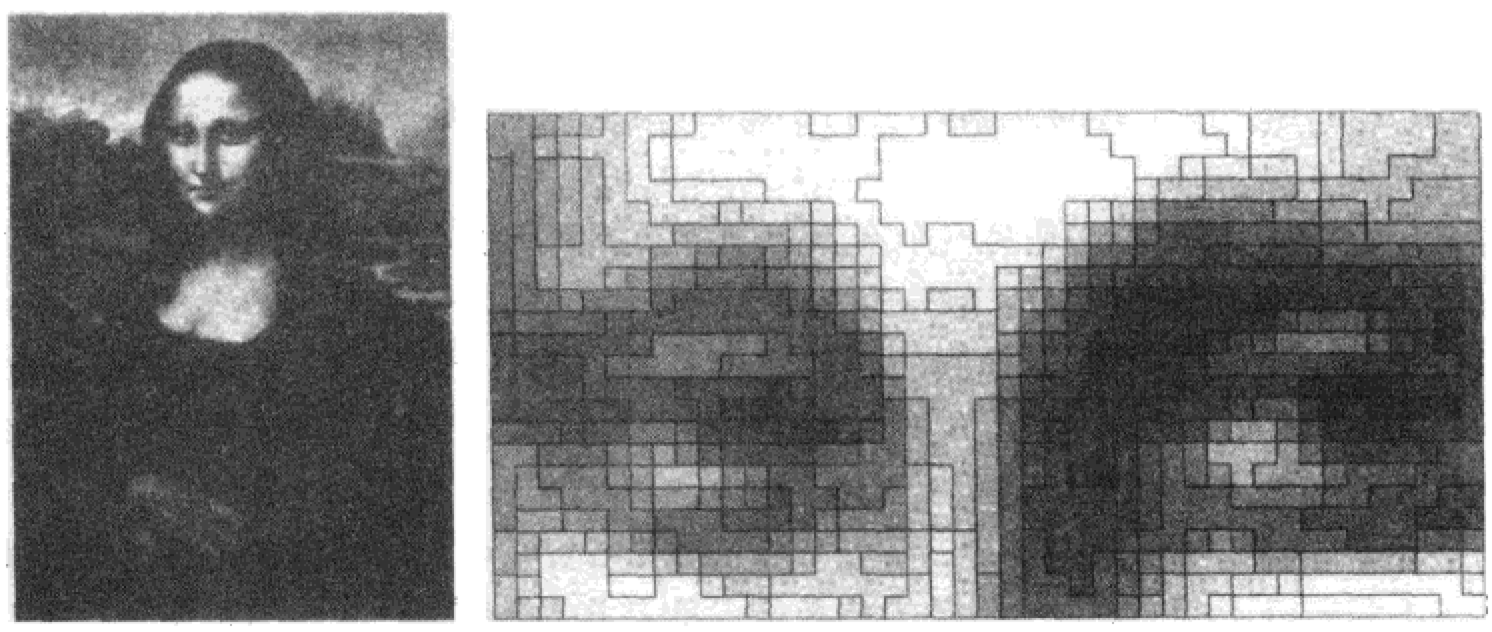


图4 蒙娜丽莎的数字重现和特写（远处观看效果更佳）

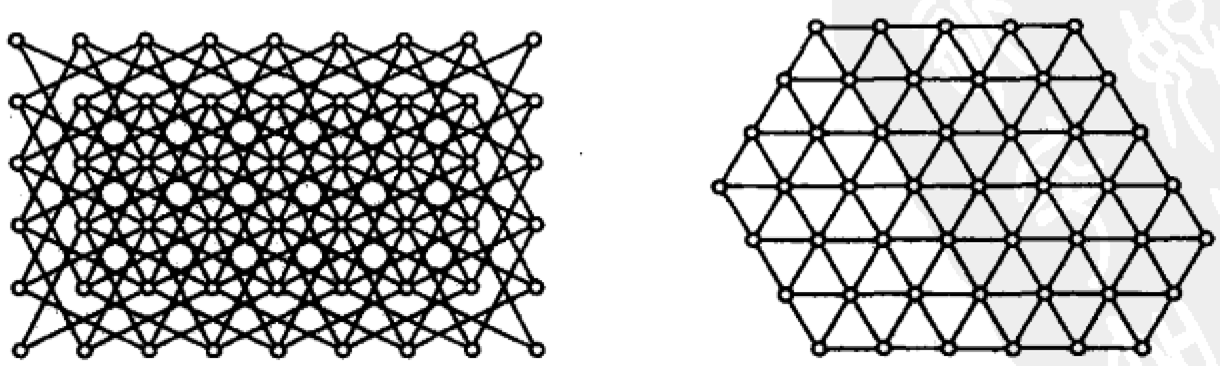
• *bi_lisa* (360, 250, 0, 360, 0, 250, 8192, 0) 是有着360+250=610个顶点和40923条边的二分图。它是另一个以列奥纳多·达·芬奇的著名绘画为出发点的图，这一次当亮度级别相差小于1/8时将行和列连接在一起。例如，边r102—c113在蒙娜丽莎“微笑”的中间靠右出现。

• *raman* (31, 23, 3, 1)相对前面的SGB图例而言，这是一个本质上不同的图。和语言、文学或其他的人类文化产物对应的图不同，这是一个基于严格数学原理的所谓“拉马努金扩展图”。图中有 $(23^3-23)/2=6072$个顶点，每个顶点的度都是32，因此边的数目是97152。顶点对应元素是模23非零的2×2矩阵的等价类。一条典型的边是(2,7;7,1)－(4,6;1,3)。拉马努金图的重要性主要是对于规模和度而言，其通常周长长而直径小。本例的周长为4，直径为1。

• *raman* (5, 37, 4, 1)与上类似，是一个有50616个顶点、151848条边，每个顶

点度都为6的正则图。其周长为10，直径为10，且正好也是个偶图。

- *random_graph* (1000, 5000, 0, 0, 0, 0, 0, 0, 0, *s*)是一个有着1000个顶点、5000条边、种子为*s*的图。它从没有边的图开始“演化”，过程是反复使用伪随机数选择顶点$0\leqslant u$，$v\leqslant 1000$，除非$u=v$或该边已经存在，否则就在图中加入边$u—v$。当$s=0$时，除了孤立顶点908外，所有其他顶点都属于阶为999的庞大连通分量。
- *random_graph* (1000, 5000, 0, 0, 1, 0, 0, 0, 0, 0) 是一个有着1000个顶点、5000条边的图，由与上例类似的演化过程而得。(事实上，该图的每条边恰好也是*random_graph*(1000, 5000, 0, 0, 0, 0, 0, 0, 0, 0)中的边)。
- *subsets* (5, 1, −10, 0, 0, 0, #1, 0)有着$\binom{11}{5}$=462个顶点，每个顶点对应着集合{0, 1, ···, 10}的包含5个元素的子集，当对应的子集不相交时，两个顶点相邻，因此该图是一个度为6、边为1386的正则图。它可以看成Petersen图的推广，Petersen图在SGB中的名字是 subsets(2, 1, −4, 0, 0, 0, #1, 0)。
- *subsets* (5, 1, −10, 0, 0, 0, #10, 0)和上例一样有462个顶点，但现在是，当相应的子集有4个共同元素时它们才是相邻的。该图是一个度为正则30的、有6930条边的图。
- *parts*(30, 10, 30, 0)是另一个有着数学背景的SGB图。其有3590个顶点，代表着将数30分成最多10个部分的一个划分，当一个划分是对另一个划分的一个组成部分进一步分解得到的时，它们是相邻的；该规则定义了31377条边。有向图*parts*(30, 10, 30, 1)的定义与此类似，但其有31377条从数目小的划分指向数目多的划分的边（如13+7+7+3→7+7+7+6+3）。
- *simplex*(10, 10, 10, 10, 0, 0) 是一个有着286个顶点、1320条边的图。顶点是方程$x_1+x_2+x_3+x_4=10$，$x_i\geqslant 0$的整数解，即将10分解为4个非负组成部分，它们也可以被看成是一个四面体的内部质心坐标。如3,1,4,2 − 3,0,4,3这样的边，将最为接近的组成成分连接在一起。
- *board*(8, 8, 0, 0, 5, 0, 0)和*board*(8, 8, 0, 0, −2, 0, 0)是一个有着64个顶点、168或280条边的图，边对应的是国际象棋中马或象的移动。调整SGB中图产生器的参数可以得到成千上万个更多的例子。例如，图5就是两个简单的*board*和*simplex*的变体，习题75将解释这些有点神秘的棋子移动规则。



board(6, 9, 0, 0, 5, 0, 0)（在6 × 9的棋盘上马的移动）　　*simplex*(10, 8, 7, 6, 0, 0, 0)（一个平切的三角网格）

图5 和棋盘相关的SGB图例

图代数　通过对已有的图进行操作也可以得到新的图。例如，如果$G=(V, E)$是一任意的图，对其求反 $\overline{G}=(V, \overline{E})$，令

$$\text{在}\overline{G}\text{中}u - v \Leftrightarrow \text{在}G\text{中}u \neq v\text{且}u \not- v \tag{35}$$

因此，非边成了边，反之亦然。注意$\overline{\overline{G}}=G$，且 $\overline{K_n}$ 中没有边。相应的邻接矩阵A和$\overline{A}$满足

$$A+\overline{A}=J-I \tag{36}$$

这里J是所有元素皆为1的矩阵，I是恒等矩阵，因此当G的阶为n时，J和$J-I$分别是J_n和K_n的邻接矩阵。

而且，每个图$G=(V, E)$都可导出一个线图（line graph）$L(G)$，其顶点是边E，若两条边有着共同的顶点则它们在$L(G)$中是邻接的。例如线图$L(K_n)$有$\binom{n}{2}$个顶点，当$n\geqslant 2$时，其度都为正则的$2n-4$（参见习题82）。当一个图的线图是k可染色时称该图是k边可染色的。

给定两个图$G=(U, E)$和$H=(V, F)$，它们的并图$G\cup H$是通过合并两个图的顶点和边得到的图（$U\cup V$，$E\cup F$）。例如，假定G和H分别是国际象棋中车和象的移动图，则$G\cup H$是皇后的移动图，其官方SGB名为

$$gunion(board(8,8,0,0,-1,0,0), board(8,8,0,0,-2,0,0),0,0) \tag{37}$$

在顶点集U和V不相交的特殊情况下，并图$G\cup H$不要求顶点的标识关于重叠关系是一致的，$G\cup H$的图形可以简单地画为G的图形紧邻着H的图形。这种特别情况称为“并置”或G和H的直和（direct sum），记为$G\oplus H$。例如，容易看出

$$K_m \oplus K_n \cong \overline{K_{m,n}} \tag{38}$$

即每个图都是其连通分量的直和。

式（38）是一般公式

$$K_{n_1} \oplus K_{n_2} \oplus \cdots \oplus K_{n_k} \cong \overline{K_{n_1,n_2,\cdots,n_k}} \tag{39}$$

的特例，对$k\geqslant 2$的完全k分图该公式都成立。但当$k=1$时，（39）却不成立。原因是存在着一个令人不快的事实：对于完全图来说，标准的图论记法是不一致的。事实上，$K_{m,n}$表示一个完全二分图，但K_n并非是完全一分图。不知为何，在这种异常下，图论学家能生活数十年而不疯狂。

另一种组合两个不相交图G和H的重要方法是形成它们的联合（join）——$G - H$，其包含了$G\oplus H$和所有形为$u - v$的边，其中$u\in U$且$v\in V$[参见A. A. Zykov，《Mat. Sbornik》24（1949），163—188，§I.3]。若G和H是不相交的有向图，它们的有向联合（directed join）$G\to H$可类似定义，其仅在$G\oplus H$上附加了单方向的从U到V的边$u\to v$。

直接将$\boldsymbol{B}$放置于$\boldsymbol{A}$的右对角线下方就可得到矩阵$\boldsymbol{A}$和$\boldsymbol{B}$的直和。

$$A\oplus B=\begin{pmatrix}A & O\\ O & B\end{pmatrix} \tag{40}$$

这里的O是全0矩阵，为使最终矩阵能排成规整的形式，还应使其有合适的行列数。由于$G\oplus H$的邻接矩阵恰是图G和H的邻接矩阵A和B的直和，因此这里使用$G\oplus H$作为图的直和记法很容易被记住。类似地，$G—H$、$G\to H$及$G\leftarrow H$的邻接矩阵分别是

$$A—B=\begin{pmatrix}A & J\\ J & B\end{pmatrix},\quad A\to B=\begin{pmatrix}A & J\\ O & B\end{pmatrix},\quad A\leftarrow B=\begin{pmatrix}A & O\\ J & B\end{pmatrix} \tag{41}$$

和（36）一样，这里J是全1矩阵。这些操作是可结合的，并可由补操作相互关联在一起。

$$A\oplus(B\oplus C)=(A\oplus B)\oplus C,\quad A—(B—C)=(A—B)—C \tag{42}$$

$$A\to(B\to C)=(A\to B)\to C,\quad A\leftarrow(B\leftarrow C)=A(A\leftarrow B)\leftarrow C \tag{43}$$

$$\overline{A\oplus B}=\overline{A}—\overline{B},\quad \overline{A—B}=\overline{A}\oplus\overline{B} \tag{44}$$

$$\overline{A\to B}=\overline{A}\leftarrow\overline{B},\quad \overline{A\leftarrow B}=\overline{A}\to\overline{B} \tag{45}$$

$$(A\oplus B)+(A—B)=(A\to B)+(A\leftarrow B) \tag{46}$$

注意，这里通过组合式（39）、（42）和（44），我们有

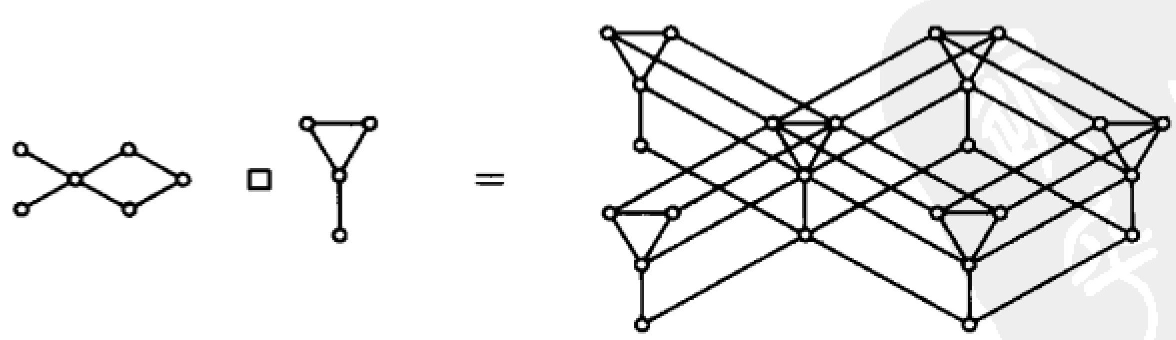
$$K_{n_1,n_2,\cdots,n_k}=\overline{K_{n_1}}—\overline{K_{n_2}}—\cdots—\overline{K_{n_k}} \tag{47}$$

这里$k\geqslant 2$。且

$$K_n=K_1—K_1—\cdots—K_1,\ K_n^{\to}=K_1\to K_1\to\cdots\to K_1 \tag{48}$$

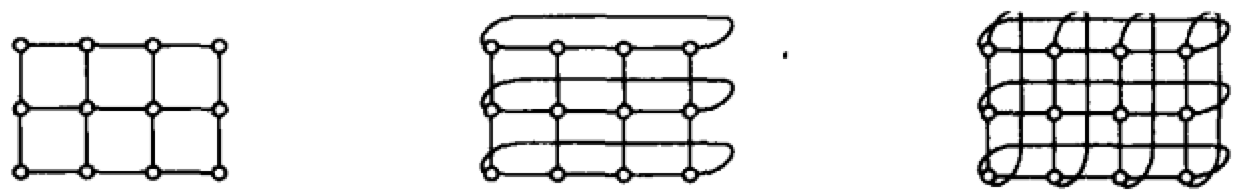
包含了n个K_1的拷贝，表明$K_n=K_{1,1,\cdots,1}$是一个完全n分图。

由于$\overline{K_m}\oplus\overline{K_n}=\overline{K_{m+n}}$且$K_m—K_n=K_{m+n}$，因此直和及联合与加法运算相似。我们也可以将图代数上的运算如乘法运算般组合在一起。例如，一个阶为m的图$G=(U, E)$和一个阶为n的图$H=(V, F)$的笛卡儿积（Cartesian product）$G\square H$是一个阶为mn的图，顶点是有序对(u, v)，其中$u\in U$且$v\in V$；当G中的$u—u'$时，边为$(u, v)—(u', v)$；当H中的$v—v'$时，边为$(u, v)—(u', v)$。换句话说，将G的每个顶点换为H的一个拷贝，将G中的边换为各相称的拷贝中相应顶点间的边就可得到$G\square H$。



(49)

如常，对于这种一般性构造，最简单的特例在实际中尤显重要。当G和H均为路径或回路时，可得“图纸之图”，即$m\times n$的网格（grid）$P_m\square P_n$、$m\times n$的圆筒（cylinder）$P_m\square C_n$和$m\times n$的圆环面（torus）$C_m\square C_n$，下面是$m=3$、$n=4$时的图示。



$P_3 \square P_4$（3×4 网格）　$P_3 \square C_4$（3×4 圆筒）　$C_3 \square C_4$（3×4 圆环面）　(50)

还有其他4种被证明是有用的图积的定义。对每种图积，顶点都是有序对(u, v)。

- 直积（direct product）$G \otimes H$，也称为G和H的合取或其“范畴乘积”，当G中有边u—u'或H中有边v—v'时，$G \otimes H$有边(u, v)—(u', v')。
- 强积（strong product）$G \boxtimes H$，其边集是$G \square H$中的边集和$G \otimes H$中的边集的并集。
- 奇积（odd product）$G \triangle H$，当G中有边u—u'或H中有边v—v'，但非同时有边时，$G \triangle H$有边(u, v)—(u', v')。
- 字典积（lexicographic product）$G \circ H$，也称为G和H的“复合”。当G中有边u—u'时，有边(u, v)—(u', v')；当H中有边v—v'时，有边(u, v)—(u, v')。

所有的这5种操作都可自然地扩充到$k \geqslant 2$个图$G_1=(V_1, E_1)$，…，$G_k=(V_k, E_k)$的乘积，其顶点为k元组$(v_1, \cdots, v_k)$，其中$v_j \in V_j$，对于$1 \leqslant j \leqslant k$。例如，当$n=3$时，考虑到复合顶点$(v_1, (v_2, v_3))$和$((v_1, v_2), v_3)$都等于$(v_1, v_2, v_3)$，因此笛卡儿集$G_1 \square (G_2 \square G_3)$和$(G_1 \square G_2) \square G_3$是同构的，因此对笛卡儿积可以略掉括号，直接写成$G_1 \square G_2 \square G_3$。有$k$个乘项的笛卡儿积的最重要例子之一是$k$立方体

$$K_2 \square K_2 \square \cdots \square K_2 \tag{51}$$

它的2^k个顶点$(v_1, \cdots, v_k)$是相邻的，当它们的海明距离为1时。

一般地，假定$v=(v_1, \cdots, v_k)$和$v'=(v'_1, \cdots, v'_k)$是k立方体中的顶点，若对于下标j，在G_j中恰好a个有v_j—v'_j，b个有$v_j=v'_j$，则有

- 当且仅当$a=1$且$b=k-1$，在$G_1 \square \cdots \square G_k$中有 v—v'；
- 当且仅当$a=k$且$b=0$，在$G_1 \otimes \cdots \otimes G_k$中有 v—v'；
- 当且仅当$a+b=k$且$a>0$，在$G_1 \boxtimes \cdots \boxtimes G_k$中有 v—v'；
- 当且仅当a是奇数，在$G_1 \triangle \cdots \triangle G_k$中有 v—v'。

由于字典积不是可交换的，因此有点不同；当且仅当v_j—v'_j，j是满足$v_j \neq v'_j$的最小下标，在$G_1 \circ \cdots \circ G_k$中有 v—v'。

习题91—102探索了图积的一些基本概念。读者还可参阅Wilfried Imrich和Sandi Klavžar（2000）所著的《Product Graphs》一书，书中除了涉及一般理论外，还包含了将一个给定的图分解为“基本”子图的算法。

***图的度序列**　若至少存在一个顶点为$\{1, 2, \cdots, n\}$的图，其中顶点k的度为d_k，则称非负整数序列$d_1 d_2 \cdots d_n$为图的（graphical）度系列。这里不妨假设$d_1 \geqslant d_2 \geqslant \cdots \geqslant d_n$。显然，对于任意的图，$d_1<n$；且任意图的度序列之和$sum=d_1+d_2+\cdots+d_n$为偶数，这是因为该值是边数的两倍。进一步，容易看出序列3311不是一个图的度序列，因此图的度序列还必须满足其他条件。那么这些条件是什么呢？

V. Havel[《Časopis pro Pěstování matematiky》80（1955），477—479]给出了一种判别一个给定序列$d_1d_2\cdots d_n$是否为图的度序列的简单方法，若满足条件的图存在，该方法还将给出图的构造。该方法从一个空的表出发，该表第k行有d_k个单元格，在要构造的图中，这些单元格是安置顶点k的邻居的“槽”。令c_j是第j列单元格的数目，则$c_1 \geqslant c_2 \geqslant \cdots$，且当$1 \leqslant k \leqslant n$时，$c_j \geqslant k$当且仅当$d_k \geqslant j$。例如，假定$n=8$且$d_1\cdots d_8=55544322$，则

1					
2					
3					
4					
5					
6					
7					
8					

(52)

是初始的表，这里$c_1\cdots c_5=88653$。Havel的想法是将顶点n的d_n个单元格和度数最大的顶点进行配对。例如，对于本例，可创建两条边8—3和8—2，此后表的形状为

1					
2					8
3					8
4					
5					
6					
7					
8	2	3			

(53)

（不创建8—1的原因是应使空单元格组成的表保持形状不变，因此每列的单元格应该从下往上进行填充。）接着，令$n \leftarrow 7$，进一步创建两条边7—1和7—5，再创建三条边6—4、6—3和5—2，这样表的单元格基本上就有一半被填充了。

1					7
2				6	8
3				6	8
4				6	
5				7	
6	2	3	4		
7	5	1			
8	2	3			

(54)

现在已将问题约简为寻找度的序列为$d_1\cdots d_5=43333$的图的问题，此时我们有$c_1\cdots c_4=5551$。在看习题103中的答案之前，希望读者能自行填充剩下的单元格。

算法H（对于给定的度的序列，生成相应的图） 给定序列$d_1 \geqslant \cdots \geqslant d_n \geqslant d_{n+1}=0$，除非$d_1\cdots d_n$不是一个图的度序列，否则算法将创建一个顶点为$\{1,\cdots,n\}$的图，其中顶点$k$的度为$d_k$，对于$1 \leqslant k \leqslant n$。数组$c_1\cdots c_{d_1}$是算法使用的辅助存储。

H1. [设置c。] 开始时令$k \leftarrow d_1$，$j \leftarrow 0$。当$k>0$时，循环执行如下操作：令$j \leftarrow j+1$；当$k>d_{j+1}$时，令$c_k \leftarrow j$且$k \leftarrow k-1$。

H2. [寻找n。] 令$n \leftarrow c_1$。若$n=0$，以成功终止；若$d_1 \geqslant n>0$，以失败终止。

H3. [对j循环。] 令$i \leftarrow 1$，$t \leftarrow d_1$且$r \leftarrow c_t$。对于$j=d_n, d_n-1, \cdots, 1$执行H4，然后

返回H2。

H4. [创建一条新边。] 令$c_j \leftarrow c_j-1$且$k \leftarrow c_t$。创建边k—n，并令$d_k \leftarrow d_{k-1}$，$c_t \leftarrow k-1$。如果$k=i$，令$i \leftarrow r+1$，$t \leftarrow d_i$且$r \leftarrow c_t$（参见习题104）。▮

如果算法H成功返回，可肯定算法构造了一个有着给定度的图；但如果算法失败返回，我们如何保证所给定的任务是不可能完成的呢？答案的关键是使用一个称作“多数化（majorization）”的重要概念。设$d_1 \cdots d_n$和$d_1' \cdots d_n'$是同一整数的两个划分（即$d_1 \geqslant \cdots \geqslant d_n$，$d_1' \geqslant \cdots \geqslant d_n'$且$d_1+\cdots+d_n=d_1'+\cdots+d_n'$），如果对于所有的$1 \leqslant k \leqslant n$有$d_1+\cdots+d_k \geqslant d_1'+\cdots+d_k'$，则称$d_1 \cdots d_n$多数化（majorize）了$d_1' \cdots d_n'$。

引理M *如果$d_1 \cdots d_n$是图的度序列且$d_1 \cdots d_n$多数化了$d_1' \cdots d_n'$，则$d_1' \cdots d_n'$也是图的度序列。*

证明：只要证明引理对只有两处不同的$d_1 \cdots d_n$和$d_1' \cdots d_n'$成立就够了。

$$d_k'=d_k-[k=i]+[k=j] \quad 这里\ i<j \tag{55}$$

原因是任何被$d_1 \cdots d_n$多数化了的序列都可以经反复执行如上的微多数化操作而得（多数化问题的详细讨论见习题7.2.1.4-45）。

条件（55）蕴涵着$d_i>d_i' \geqslant d'_{i+1} \geqslant d_j'>d_j$。因此任何度的序列为$d_1 \cdots d_n$的图都包含有一个顶点$v$，$v$满足$v$—$i$且$v \not- j$。删除边$v$—$i$和增加边$v$—$j$'就可产生所要的度的序列为$d_1' \cdots d_n'$的一个新图。▮

推论H *当$d_1 \cdots d_n$是图的度序列时，算法H成功返回。*

证明：不妨假定$n>1$。假设G是顶点为$\{1, \cdots, n\}$，度的序列为$d_1 \cdots d_n$的任意一个图，G'是一个由$\{1, \cdots, n-1\}$所诱导的子图，换句话说，移去顶点n和相关联的d_n条边后得到的图。G'的度序列$d_1' \cdots d'_{n-1}$可由$d_1 \cdots d_{n-1}$中d_n个的值减去1再重排成非递减序列而得。由定义可知，$d_1' \cdots d'_{n-1}$是图的度序列。由策略H3和H4所生成的新的度序列$d_1'' \cdots d''_{n-1}$可被每个如此得到的$d_1' \cdots d'_{n-1}$所多数化，原因是它将最大的d_n个合适元素各减少了1。因此新的序列$d_1'' \cdots d''_{n-1}$是图的度序列。在算法H中，令$d_1 \cdots d_{n-1} \leftarrow d_1'' \cdots d''_{n-1}$，因此对$n$进行归纳即可证明推论。▮

算法H的运行时间大约和所生成的边的数目成比例，即是n^2阶的。习题105给出了一种更快的方法，其可在$O(n)$步内判别给定的序列$d_1 \cdots d_n$是否是图的度序列（无需构造任何图）。

图的超越 当一个图或一个有向图的顶点或边使用另外的数据修饰时，该图就称为网络（network）。例如，*words*(5757, 0, 0, 0)上的每个顶点有一个附加秩，其对应的是相应的5字母单词的普遍程度。*plane_lisa*(360, 250, 15, 0, 360, 0, 250, 0, 0, 229500)上的每个顶点有附加的0～15的像素密度。*board*(8, 8, 0, 0, −2, 0, 0)的每个弧都有附加长度，其表示棋盘上一个棋子的移动距离，如从一个角落移动到另一个角度的长度为7。斯坦福图库还包括另外一些、这里没有提及的图生成器，它们主要被用来生成一些有趣的结构。

• *miles*（128, 0, 0, 0, 0, 127, 0）是一个有128个顶点的网络，对应于前面提及的图*plane_miles*中的北美城市。但和*plane_miles*不同的是，*miles*是一个

有$\binom{128}{2}$条边的完全图。边的长度为整数，其表示在1949年一辆轿车或卡车从一个给定城市到达另一个给定城市需要跨越的距离。例如，在*miles*网络中，从“Vancouver, BC”到“West Palm Beach, FL”的距离是3496英里。

- *econ*(81, 0, 0, 0)是一个有81个顶点和4902条弧的网络。其顶点表示美国经济部门，边表示1985年间从一个部门流向另一个部门的以百万美元计的资金。例如，从Apparel流向Household furniture的流量是44，表示该年度家居业向服装业支付了$44 000 000的资金。流入每个顶点的资金总和等于流出每个顶点的资金总和。弧只在流非零时出现。称为Users的特殊顶点所接受的流量代表对产品的总需求；图中有些最终用户的流量是非负的，这是因为进口货物是由政府经济学家处理的。
- *games*（120, 0, 0, 0, 0, 0, 128, 0）是一个有着120个顶点和1276条弧的图。顶点表示美国大学足球队，边表示在1990赛季中两队曾经交锋，边标记为得分值。例如，弧Stanford→California标记为27，而弧California→Stanford标记为25，这是因为在1990年11月17日斯坦福红雀队以27比25打败了贝克莱金熊队。
- *risc*（16）是一个完全不同类型的网络。它有3240个顶点、7878条弧，定义了一个有向无循环图或俗称的“dag”，即一个不包含有限回路的有向图。顶点代表有布尔值的门；如弧Z45→R0:7~表示Z45门的输出值是R0:7~门的一个输入。每个门有一个类型码（AND、OR、XOR、NOT、锁存或外部输入）；每条弧有一个长度，表示延迟量。网络包含了一个微型RISC芯片的完整逻辑电路，它可执行由16个16位寄存器所支配的简单指令。

关于SGB生成器的所有完全细节都可以在作者的书籍《The Stanford GraphBase》（New York：ACM Press，1993）中找到，书中还包括了数十个简短的程序例子以说明如何对生成器生成的图或网络进行操作。例如，名为LADDERS的程序给出了如何寻找一个5字母单词和另一个5字母单词间的最短路径。称为TAKE_RISC的程序则演示了如何基于*risc*(16)中门电路所构造的网络的行为对纳米计算机进行步速仿真。

超图 图和网络非常迷人，但不管如何，它们还不是故事的全部。许多重要的组合算法的设计使用了超图（hypergraph），超图允许边关联任意的顶点集，它比图更常用。

例如，超图可有7个顶点，用非零二进制串$v=a_1a_2a_3$标识，有7条边，用括号括起来的非零二进制串$e=[b_1b_2b_3]$标识，定义$v\in e$当且仅当$(a_1b_1+a_2b_2+a_3b_3)\bmod 2=0$。此时每条边恰好包含了3个顶点。

$$[001]=\{010,100,110\};\quad [010]=\{001,100,101\};\quad [011]=\{011,100,111\}$$
$$[100]=\{001,00,011\};\quad [101]=\{010,101,111\} \qquad (56)$$
$$[110]=\{001,110,111\};\quad [111]=\{011,101,110\}$$

由对称性可知，每个顶点恰好属于3条边。(为了与通常的图中的边相区别，包含3个及以上顶点的边有时称为“超边”，但称呼它们为“边”也不会有问题)。

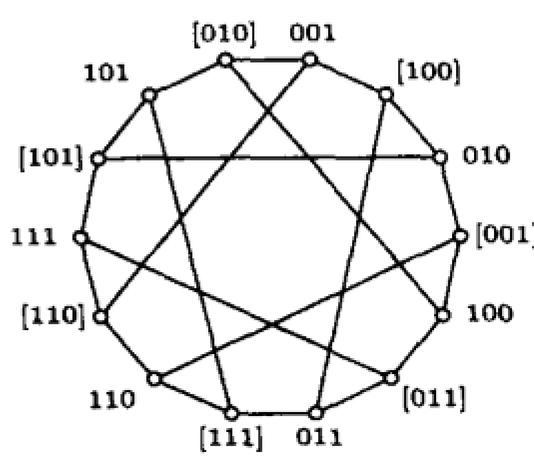
如果图的每条边恰好包含r个顶点，称一个超图是r一致的（r-uniform）。由此，（56）是一个3一致的超图，2一致超图是指通常的图。完全r一致的超图$K_n^{(r)}$有n个顶点和$\binom{n}{r}$条边。

大多数图论的基本概念都可以自然地扩展到超图。例如，如果$H=(V, E)$是一个超图，且$U \subseteq V$，则由U导出的子超图$H|U$的边为$\{e | e \in E 且 e \subseteq U\}$。$r$一致超图$H$的补$\overline{H}$的边是不在$H$中的$K_n^{(r)}$的边。一个超图的$k$着色是对超图顶点进行颜色赋值以使没有一条边是单色的，等等。

由于同一性质可以有不同的阐述方式，超图有着许多不同的名字。例如，因为每条边都是V的子集，因此本质上每个超图$H=(V, E)$都是一个集合簇（family of sets）。一个3一致的超图也被称为一个三元系统（triple system）。一个超图也等价于一个0和1的矩阵$\boldsymbol{B}$，行对应顶点v，列对应边e，矩阵的v行和e列包含值$b_{ve}=[v \in e]$。矩阵$\boldsymbol{B}$被称为H的关联矩阵（incidence matrix），当$v \in e$时我们称“v和e关联”。进一步，一个超图和一个顶点集为$V \cup E$的偶图（bipartite graph）是等价的，当超图中v和e关联时，偶图中有边$v — e$。超图是连通的当且仅当相应的偶图是连通的。超图中一个长为k的回路（cycle）对应相应偶图中一个长为$2k$的回路。

例如，（56）中的超图可以由等价的关联矩阵或等价的偶图定义如下：

	[001]	[010]	[011]	[100]	[101]	[110]	[111]
001	0	1	0	1	0	1	0
010	1	0	0	1	1	0	0
011	0	0	1	1	0	0	1
100	1	1	1	0	0	0	0
101	0	1	0	0	1	0	1
110	1	0	0	0	0	1	1
111	0	0	1	0	1	1	0



(57)

它包含28个长为3的回路，如

$$[101]—101—[010]—001—[100]—010—[101] \tag{58}$$

交换边和顶点的角色并保留关联关系不变可以得到超图H的对偶H^{T}。换句话说，其对应着关联矩阵的转置。要注意的是，一个r正则图的对偶是一个r一致超图。

关联矩阵和偶图也可和超图对应，由于矩阵中不同列可能相等，因此超图中一些边的出现可能多于一次。不含重边的超图$H=(V, E)$对应的是另一种组合对象，即布尔函数。比如说，顶点集V为$\{1, 2, \cdots, n\}$，函数

$$h(x_1, x_2, \cdots, x_n) = [(j \mid x_j = 1) \in E] \tag{59}$$

描述了超图H的边。例如，布尔公式

$$(x_1 \oplus x_2 \oplus x_4) \wedge (x_2 \oplus x_3 \oplus x_5) \wedge (x_3 \oplus x_4 \oplus x_6) \wedge (x_4 \oplus x_5 \oplus x_7) \wedge (x_5 \oplus x_6 \oplus x_1) \wedge (x_6 \oplus x_7 \oplus x_2) \wedge (x_7 \oplus x_1 \oplus x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \tag{60}$$

是另一种刻画（56）和（57）中超图的方式。

事实上，可以从多个不同方面考察组合对象这一事实可能会令人发怵，但也非常有用，因为其提示了以不同方式解决等价问题的方法。从不同的视角看待一个问题，我们的大脑自然会以不同的方式运转。有时通过思考如何操作矩阵的行和列会更好地透彻了解问题；有时对顶点和路径进行想象，或对空间中的点簇进行形象思维，我们会得到进展；有时布尔代数是解决问题的关键。有时我们被困于一个领域，而另一个领域中存在的方法则可能正是我们的救星。

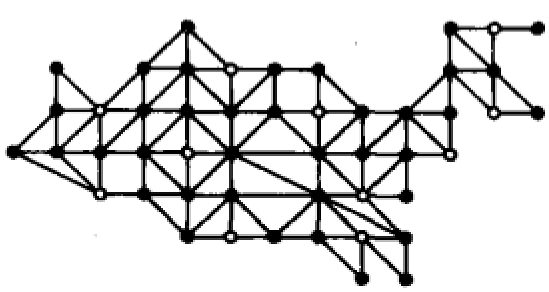
覆盖和独立 如果$H=(V, E)$是一个图或超图，如果图中每条边至少包含顶点集U中的一个元素，则称U覆盖（cover）H。如果没有一条边完全包含于顶点集W之中，则W在H中被认为是独立的（independent）或“稳定的”。

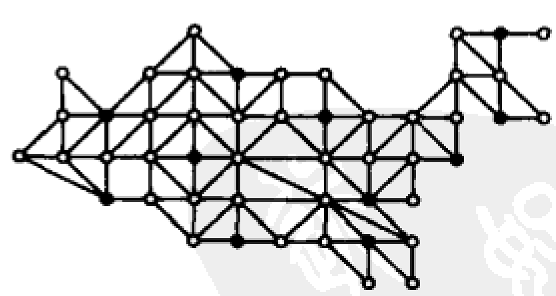
从关联矩阵的角度看，覆盖是行的集合，其每列的和非零。对于H是图这种特殊情况，矩阵的每列仅包含2个1，因此图的独立集对应于互为正交的行集合，即任何两个不同行的点积的值为0。

这些概念就像同一硬币的对立面。如果U覆盖H，则在H中$W=V\setminus U$是独立的；反之，如果在H中W是独立的，则$U=V\setminus W$覆盖H。上述两个语句同样说明了导出超图$H|W$是没有边的。

最早大概是Claude Berge所记录[《Proc. National Acad. Sci.》43（1957），842－844]的覆盖和独立集间的对偶关系有点似是而非的感觉。尽管在逻辑上显然成立并易于验证，但在直觉上却有点出乎人的意料。在看着一个图并尝试找出其最大独立集时，我们的思路和看着同一个图并尝试找出其最小的顶点覆盖是完全不同的；然而两者的最终目标却是一样的。

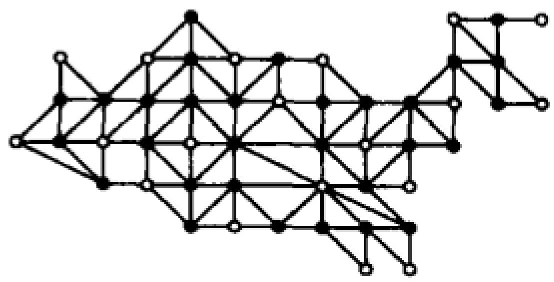
若对于所有$u\in U$，$U\setminus u$不是覆盖集，则覆盖集U是极小的（minimal），若对于所有$w\notin W$，$W\cup w$不是独立的，则独立集W是极大的（maximal）。例如，下面是表示美国连续疆土（17）的含49个顶点的极小覆盖和对应的极大独立集。


有38个顶点的极小顶点覆盖

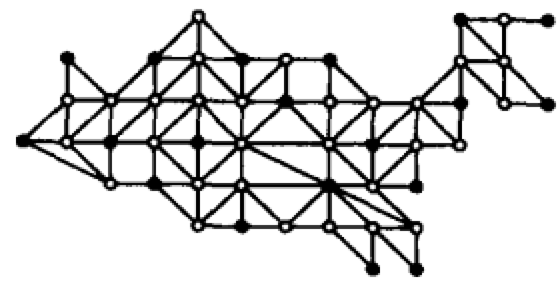

有11个顶点的极大独立集

(61)

一个覆盖，如果其规模是可能的覆盖中最小的，则称其为最小覆盖（minimum）；一个独立集，如果其规模是可能的独立集中最大的，则称其为最大独立集（maximum）。例如，对于图（17），我们可以做得比（61）更好。



有30个顶点的最小顶点覆盖



有19个顶点的最大独立集

(62)

注意这里“极小”和“最小”的微妙区别：通常（但和大多数的字典相反），在进行组合算法的研究时，对于局部最好的组合格局，通常使用“极”，如“极小”或“极优”来称谓。相应地，“最”词如“最小”或“最优”，被留着用于来称谓全局最好，即考虑了所有可能性的组合格局。对于任何优化问题，通过爬山迭代直到抵达山顶，容易找出的往往是极优的答案。要找出真正最优的答案，通常非常困难。例如，在7.9节我们将看到，在给定图中寻找最大独立集的问题属于被称为NP完全的难题类。

尽管面临的问题是NP完全的，我们也无需绝望。本章好几个地方都将讨论寻找最小覆盖的技术，这些技术对于较小的问题都做得很好。例如对于（62），只需对2^{49}种可能性的很小一部分进行检查就可以了，因此在1秒内就可得到问题的答案。进一步，一些NP完全问题的特例通常要比一般情况简单。在7.5.1节我们将看到，对于任意偶图或是一个图的对偶的任何超图，我们都可以快速找到最小顶点覆盖。在7.5.5节，我们将研究寻找最大匹配（matching）的有效方式，最大匹配就是给定图的线图（line graph）的最大独立集。

由于经常出现对一个独立集的规模进行最大化这一问题，对此问题这里给出一种特殊记法：如果H是一个超图，则数值

$$\alpha(H) = \max\{|W| \mid W\text{是}H\text{中顶点的独立集}\} \tag{63}$$

称为H的独立数（independence number）或稳定数（stability number）。类似地，

$$\chi(H)=\min\{k \mid H\text{是}k\text{可着色的}\} \tag{64}$$

称为H的着色数（chromatic number）。注意，$\chi(H)$是由独立集得到的H的最小覆盖的大小，因为接受任意特定颜色的顶点按照这里的定义必须是独立的。

若H是通常的图，$\alpha(H)$和$\chi(H)$的定义也适用，但通常写成$\alpha(G)$和$\chi(G)$。图有另一种重要的数，称为团数（clique number）

$$\omega(G) =\max\ \{|X| \mid X\ \text{是}G\text{中的团}\} \tag{65}$$

其中“团”是相互邻接的顶点。显然

$$\omega(G) = \alpha(\overline{G}) \tag{66}$$

这是因为G的一个团是其补集的独立集。类似地，$\chi(\overline{G})$可以看成最小规模的“团覆盖”，它是恰好覆盖所有顶点的团的集合。

本节前面已经提及了数个“精确覆盖问题”的实例，但没有确切精确这些问题表示什么意思。现在是给出定义的时候了：给定超图H的关联矩阵，H的一个精

确覆盖是和为(11···1)的行的集合。换句话说，一个精确覆盖是一系列顶点的集合，其和每条“超边”的接触恰好是一次；通常的覆盖则仅需要和每条“超边”接触至少一次。

习 题

1. [25] 设$n=4m-1$。对于数{1，1，…，n，n}构造朗福德对排列，满足性质：把其第一个‘$2m-1$’换成‘$4m$’，并把‘$2m-1$ $4m$’附加到右边，就能获得$n=4m$的朗福德对排列。提示：把$m-1$个偶数$4m-4$，$4m-6$，…，$2m$放在左边。

2. [18] 问n为何值时，{0，0，1，1，…，$n-1$，$n-1$}可以排列成朗福德对？

3. [22] 若把{0，0，1，1，…，$n-1$，n-1}排成环形，而非一列，且让两个k之间的距离为k。问是否可得完全不同于习题2的解？

4. [M20] (T. Skolem，1957。)试解释，只需将所有的a和所有的b从左到右依次分别用0，1，2…替换，则由1.2.8节的习题36中的斐波那契串$S_\infty=babbababbabba\cdots$就可直接构造出一个由所有非负整数组成的朗福德对的无限序列0012132453674…。

▸ 5. [HM22] 随机选取{1，1，2，2，…，n，n}的一个排列，对于给定的k，两个k刚好相隔k个位置的概率是多少？试使用(1)中的公式估计朗福德对的数目L_n。

▸ 6. [M28] (M. Godfrey，2002。)令 $f(x_1,\cdots,x_{2n})=\prod_{k=1}^{n}\left(x_k x_{n+k}\sum_{j=1}^{2n-k-1}x_j x_{j+k+1}\right)$。

a）试证明 $\sum_{x_1,\cdots,x_{2n}\in\{-1,+1\}}f(x_1,\cdots,x_{2n})=2^{2n+1}L_n$。

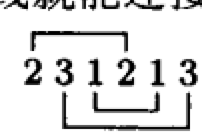
b）解释如何使用$O(4^n n)$步操作计算出和值。在运算中需要多少个精确比特位？

c）利用下述等式可获8倍的计算加速，试证下述等式成立。

$$f(x_1,\cdots,x_{2n})=f(-x_1,\cdots,-x_{2n})=f(x_{2n},\cdots,x_1)=f(x_1,-x_2,\cdots,x_{2n-1},-x_{2n})$$

7. [M22] 证明{1，1，…，16，16}的任一个朗福德对中皆存在某一点，当从左往右阅读时有7个不完整的数对。

8. [23] 最简单的朗福德序列不仅是均衡的，它还是个平面图（planar），即所有数对均可以不用(2)中出现的交叉线就能连接在一起。



试找出$n\leqslant 8$的所有是平面图的朗福德对。

9. [24]（朗福德三元组，Langford triples）将{1，1，1，2，2，2，…，9，9，9}排成一行，要求连续的3个k相隔k个位置（$1\leqslant k\leqslant 9$）有多少种方式？

10. [M20] 解释怎样从图1构造一个魔方（magic square）。（将每一张卡片用1和16间的一个数标记，对卡片进行排列，使每行、每列以及主对角线卡片上的数字加起来的和都等于34）。

11. [20] 将（5）扩展为一个“希伯来-希腊-拉丁”方阵，即在每一个由两个字母组成的元素后面添加一个希伯来字母{א,ב,ג,ד}，使字母对（拉丁，希腊）、

（拉丁，希伯来）或（希腊，希伯来）不在多于一个地方出现。

▸12. [M21] (L. Euler.) 设$L_{ij}=(i+j) \bmod n$，对于$0\leqslant i$，$j<n$是模n的整数加法表。证明当且仅当n是奇数，存在与L正交的拉丁方阵。

13. [M25] 一个10×10的方阵可以被分成4个5×5的方阵。在一个由数字{0，1，…，9}形成的10×10的拉丁方阵中，当其左上角恰好有k个元素大于等于5时，称其有k个“干扰者”(intruders)（参见习题14(e)中$k=3$时的例子）。证明该方阵除非至少有3个干扰者否则没有正交配偶。

14. [29] 找出下列拉丁方阵的所有正交配偶。

(a)	(b)	(c)	(d)	(e)
3145926870	2718459036	0572164938	1680397425	7823456019
2819763504	0287135649	6051298473	8346512097	8234067195
9452307168	7524093168	4867039215	9805761342	2340178956
6208451793	1435962780	1439807652	2754689130	3401289567
8364095217,	6390718425,	8324756091,	0538976214,	4012395678
5981274036	4069271853	7203941586	4963820571	5678912340
4627530981	3102684597	5610473829	7192034658	6789523401
0576148329	9871546302	9148625307	6219405783	0195634782
1730689452	8956307214	2795380164	3471258906	1956740823
7093812645	5643820971	3986512740	5027143869	9567801234

15. [50] 找出3个两两正交的10×10的拉丁方阵。

16. [48] (H. J. Ryser，1967.) 如果一个拉丁方阵有n行、n列、n个符号，则称其为“n阶的”。试问每个奇数阶拉丁方阵是否都有一个横截？

17. [25] 设L是一个由L_{ij} ($0\leqslant i$，$j<n$) 构成的拉丁方阵。证明(a)找出L的所有横截和(b)找出L的所有正交配偶均是普通的精确覆盖问题的特例。

18. [M23] 如果字符串$x_1x_2\cdots x_N$的每个元素x_j都属于n个元素组成的集合{0，1，…，$n-1$}，则称这个字符串是“n元（n-ary）”的。如果两个字符串$x_1x_2\cdots x_N$和$y_1y_2\cdots y_N$的N个字符对$(x_j, y_j)(1\leqslant j\leqslant N)$都是不同的，则称这两个字符串正交。（因此，如果两个n元字符串的长度N超过了n^2，它们就不可能是正交的。）一个由n元字符串组成的m行、n^2列矩阵，若其每一行均与其余各行正交，则称这个矩阵为阶为n、深度为m的正交阵列（orthogonal array）。

试找出深度为m的正交阵列和$m-2$个相互正交的拉丁方阵的列表间的对应关系。与习题11对应的正交阵列是什么？

▸19. [M25] 接习题18，证明阶大于1、深度为m的正交阵列仅当$m\leqslant n+1$时才可能存在。解释当n是素数p时这个上限是可达的。写出$p=5$时的一个例子。

20. [HM20] 如果一个正交阵列的每一个元素k都被替换为$e^{2\pi ki/n}$，行之间的关系即是通常意义上的正交关系（它们的点积为零）。

▸21. [M21] 一个几何网（geometric net）是一个由点和线组成的系统，它遵从下面三条公理。

i）每一条线都是点的集合。

ii）不同线间最多只有一个公共点。

iii）如果p是一个点，L是一条线，且$p\notin L$，则只存在唯一的一条线M满足$p\in M$且$L\cap M=\varnothing$。

如果$L\cap M=\varnothing$，则称L平行于M，记为$L\parallel M$。

a）证明一个几何网中的所有线均可化归为等价类，两条线在同一个类中当且仅当它们相等或平行。

b）如果至少存在两个平行线类，则每条线和同一个类中的其他线有着同样数目的交点。

c）此外，如果有至少三个类，则存在数m、n，使所有点刚好属于m条线且所有线恰好包含n个点。

▶22. [M22] 解释每一个正交阵列都可以看作一个几何网。逆命题是否成立？

23. [M21]（纠错编解码，error-correcting code）两个字符串$x=x_1\cdots x_N$和$y=y_1\cdots y_N$的海明距离$d(x, y)$是指$x_j\neq y_j$的位置j的数目。一个有n个信息位和r个校验位的b元码是b^n个字符串$x=x_1\cdots x_{n+r}$ $(0\leqslant x_j<b,\ 1\leqslant j\leqslant n+r)$的一个集合$C(b,n,r)$。传输一个码字$x$并接收到消息$y$，$d(x,y)$即是传输错误的数目。如果无论何时收到的消息$y$有$d(x,y)\leqslant t$，我们都可以从$y$重构出$x$的值，则称这种编码为$t$纠错码。编码的距离是指，对于所有码字对$x\neq x'$，$d(x,x')$的最小值。

a）证明一个编码是t纠错码当且仅当它的距离超过$2t$。

b）证明一个有2个信息位和2个校验位的单纠错b元码等价于一对正交的b阶拉丁方阵。

c）进一步，一个距离为$r+1$的编码$C(b,2,r)$等价于r个相互正交的b阶拉丁方阵组成的集合。

▶24. [M30] 由一个具有N个点和R条线的几何网可以方便地构造出二元码$C(2, N, R)$，码字$x_1\cdots x_N x_{N+1}\cdots x_{N+R}$用奇偶校验位定义

$$x_{N+k}=f_k(x_1,\cdots,x_N)=(\Sigma\{x_j \mid 点\ j位于线k上\})\bmod 2$$

a）如果这个网有m个平行线类，证明这个编码的距离等于$m+1$。

b）找出一个高效的方法使用这个编码纠正t个错误，假定$m=2t$。阐述当$N=25$，$R=30$，$t=3$时的译码过程。

25. [27] 找出一个拉丁方阵，它的行和列都是5个字母的单词（要完成本习题，你需要查阅大词典）。

▶26. [25] 构造一个仅仅包含5个字母的单词的有意义的英文句子。

27. [20] 有多少个SGB中的单词刚好包含k个不同的字母，对于$1\leqslant k\leqslant 5$？

28. [20] 是否存在SGB中单词对应的向量对，向量间的每个元素相差±1？

29. [20] 找出所有是回文的（palindromes，即从左往右看与从右往左看是一样的）或镜像的SGB单词对（如`regal lager`）。

▶30. [20] `first`的字母从左到右是按字母升序的。在词典中第一个这样的5个字母的词是什么？最后一个是什么？

31. [21]（C. McManus.）找出所有三个SGB单词构成的集合，要求它们是算术级数[⊖]，且在任一固定的位置上没有相同的字母。（一个这样的例子是`{power`，

⊖ 等差数列。——译者注

slugs, visit}。)

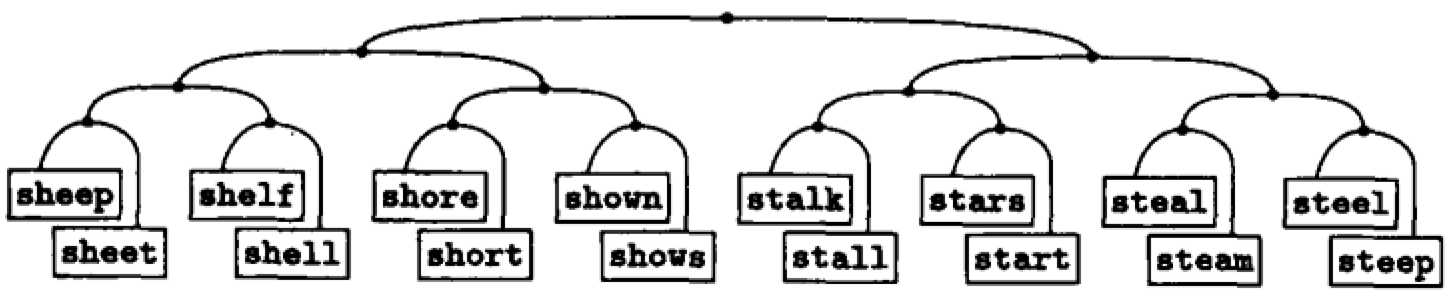
32. [23] 英语语言是否包含任何10个字母的单词$a_0a_1\cdots a_9$，其中$a_0a_2a_4a_6a_8$和$a_1a_3a_5a_7a_9$都是SGB单词？

33. [20] (Scot Morris.) 完成下列26个有趣的SGB单词的列表：

about, bacon, faced, under, chief, …, pizza

▸ 34. [21] 对每一个不包含字母y的SGB单词，把它的元音字母{a,e,i,o,u}换作1，其他字母换作0，可得一个5位的二进制数。这32个不同的二进制数分别可以来自哪些常见的单词？

▸ 35. [26]下图是从WORDS(1000)中精心挑选的16个元素形成的分支模式(branching pattern)，它是一个由字母s开头的单词所组成的完全二分查找树(binary trie)。



但即便考虑完整的集合WORDS(5757)，以a开头的单词也没有这种模式。对于给定的n，字母表中的哪些字母可以被用作WORDS(n)中的16个单词的起始字母，使这16个单词能形成一个完整的二分查找树？

36. [M17] 解释出现在立方体(10)中的对称性。另外再解释一下仅改变两个词{stove, event}就可以得到另外两个类似的立方体。

37. [20] 图$words(5757, 0, 0, 0)$的哪一个顶点度数最大？

38. [22] 请使用(14)中的变换规则，仅用三步就将tears变成smile，不要借助计算机。

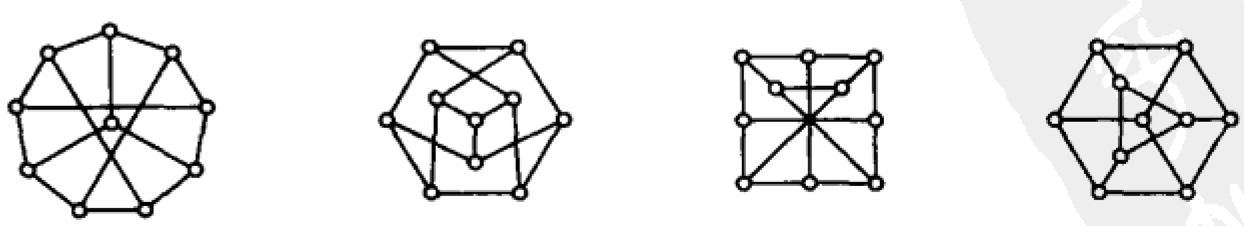
39. [M00] $G\backslash e$是否是G的一个导出子图？它是否是一个生成子图呢？

40. [M15] 对于一个图$G=(V, E)$，当$|V|=n$且$|E|=e$时，有多少个(a)生成子图(b)导出子图？

41. [M10] 对哪些整数n，有(a)$K_n=P_n$？(b)$K_n=C_n$？

42. [15] (D. H. Lehmer) 设G是一个有13个顶点的图，每个顶点的度都为5。给出G的一个非平凡的描述。

43. [23] 下列诸图中是否有和Petersen图相同的图？



44. [M23] Chvátal图中有多少个对称关系？(参见图2(f)。)

45. [20] 给出一种简单的将平面图(17)用4种颜色着色的方法。3种颜色够用吗？

46. [M25] 设G是一个顶点数$n\geqslant 3$的平面图。如果在它的不相邻的顶点间不能再添加任何一条边而不和已有的边相交，则称它是最大平面图。

a）证明最大平面图将平面划分为若干区域，其中每个区域的边界上恰有三个顶点。（有一个区域是所有点的集合，处于图的外部。）

b）最大平面图G恰有$3n-6$条边。

47. [M22] 证明完全二分图$K_{3,3}$不是平面图。

48. [M25] 通过解释文中描述的过程不会将两个相邻顶点着为同一个颜色，来完成定理B的证明。

49. [18] 画出最多6个顶点的所有立方图（cubic graph）。

50. [M24] 找出所有可恰好使用24种方式进行3着色的二分图。

▸51. [M22] 给定一个如习题21所描述的几何网，请构造一个二分图，其顶点分别是网中的点p和线L，且p—L当且仅当$p\in L$。这个图的周长是多少？

52. [M16] 找出一个简单的图的直径和周长间的不等关系。（如果周长较大，直径可以有多小？）

53. [15] `world`和`happy`中的哪一个词是属于图$words(5757, 0, 0, 0)$中的庞大分量？

▸54. [21] 在图(17)中的49个邮政编码按字典序是`AL`，`AR`，`AZ`，`CA`，`CO`，`CT`，`DC`，`DE`，`FL`，`GA`，`IA`，`ID`，`IL`，`IN`，`KS`，`KY`，`LA`，`MA`，`MD`，`ME`，`MI`，`MN`，`MO`，`MS`，`MT`，`NC`，`ND`，`NE`，`NH`，`NJ`，`NM`，`NV`，`NY`，`OH`，`OK`，`OR`，`PA`，`RI`，`SC`，`SD`，`TN`，`TX`，`UT`，`VA`，`VT`，`WA`，`WI`，`WV`，`WY`。

a）假设两个州的邮编有一个位置的字母是一样的（即`AL-AR-OR-OH`等）则认为它们是相邻的。这个图的连通分量是什么？

b）约定`XY`→`YZ`为边的方向例如，`AL`→`LA`→`AR`等，现由邮编可形成一个有向图等。这个有向图的强连通分量是什么？（参见2.3.4.2节。）

c）除了(17)给出的邮政编码外，`AA`，`AE`，`AK`，`AP`，`AS`，`FM`，`GU`，`HI`，`MH`，`MP`，`PW`，`PR`，`VI`也是美国的邮政编码。试使用所有62个邮政编码重新思考问题(b)。

55. [M20] 在完全k分图$K_{n_1, n_2, \cdots, n_k}$中有多少条边？

▸56. [M10] 判断正误：多重图是一个图，当且仅当相应的有向图是简单图。

57. [M10] 判断正误：，当且仅当$d(u, v)<\infty$或$d(v, u)<\infty$，顶点u和v在有向图的同一个连通分量中。

58. [M17] 描述(a)所有度为2的正则图；(b) 描述所有度为2的多重图。

▸59. [M23] 一个n阶竞赛图（tournament）是一个有n个顶点的有向图，它恰有$\binom{n}{2}$条弧，即对每对不同的顶点$\{u, v\}$有$u\to v$或$v\to u$。

a）证明每个竞赛图包含一个有向生成路径（oriented spanning path）$v_1\to\cdots\to v_n$。

b）考虑顶点$\{0, 1, 2, 3, 4\}$上的竞赛图，当且仅当$(u-v)\bmod 5\geqslant 3$，$u\to v$。它有多少个有向生成路径？

c）$K_n^{\rightarrow}$是否是仅有的具有唯一有向生成路径的n阶竞赛图？

▶60. [M22] 设u是竞赛图中具有最大出度的顶点，而v是任何其他的顶点。证明$d(u, v)\leqslant 2$。

61. [M16] 构造一个有向图，从顶点1到顶点2有k个长度为k的路线。

62. [M21] 一个有向置换图（permutation digraph）是一个有向图，它的每一个顶点的出度和入度都为1；因此它的分量都是有向回路。如果它有n个顶点和k个分量，若$n-k$为偶数则称它为偶图，若$n-k$为奇数则称它为奇图。

a）设G是一个有向图，用邻接矩阵$\boldsymbol{A}$表示。证明G的有向生成置换图的数目是per $\boldsymbol{A}$，即$\boldsymbol{A}$的积和式。

b）用有向生成置换图解释$\boldsymbol{A}$的行列式det $\boldsymbol{A}$。

63. [M23] 设G是一个周长为g的图，其中每一个顶点都有至少d个邻居。证明G有至少N个顶点，N满足

$$N=\begin{cases}1+\sum_{0\leqslant k<t} d(d-1)^k, & \text{如果}g=2t+1\\ 1+(d-1)^t+\sum_{0\leqslant k<t} d(d-1)^k, & \text{如果}g=2t+2\end{cases}$$

▶64. [M21] 接习题63，解释仅存在一个周长为4、最小度为d且阶为$2d(d\geqslant 2)$的图。

▶65. [HM31] 假设图G周长为5，最小度为d，有$N=d^2+1$个顶点。

a）证明G的邻接矩阵$\boldsymbol{A}$满足等式$\boldsymbol{A}^2+\boldsymbol{A}=(d-1)\boldsymbol{I}+\boldsymbol{J}$。

b）因为$\boldsymbol{A}$是一个对称矩阵，它有N个正交的特征向量x_j和相应的特征值λ_j，即$\boldsymbol{A}x_j=\lambda_j x_j$ $(1\leqslant j\leqslant N)$。证明每一个$\lambda_j$要么是$d$，要么是$(-1\pm\sqrt{4d-3})/2$。

c）解释如果$\sqrt{4d-3}$是无理数，则$d=2$。提示：$\lambda_1+\cdots+\lambda_N=\mathrm{trace}(\boldsymbol{A})$（矩阵$\boldsymbol{A}$的迹）$=0$。

d）解释如果$\sqrt{4d-3}$是有理数，则$d\in\{3, 7, 57\}$。

66. [M30] 接习题65，若$d=7$，构造一个这样的图。

67. [M48] 是否存在一个度为57，阶为3250，且周长为5的正则图？

68. [M20] 一个有n个顶点的图G有多少个不同的邻接矩阵？

▶69. [20] 扩展(31)，给定一个以SGB格式表示的图，怎样计算所有顶点v的出度`ODEG`(v)和入度`IDEG`(v)。

▶70. [M20] 当算法B成功对一个有m条弧和n个顶点的图2着色时，给出算法中每一步的执行频率。

71. [26] 在`MMIX`计算机上实现算法B，使用`MMIXAL`汇编语言。假定当你的程序开始时，寄存器v0指向第一个顶点节点，寄存器n保存了顶点的数目。

▶72. [M22] 在B6步设置`COLOR`(v)时，称u为v的父节点（parent）；在B3步设置`COLOR`(w)时，说w没有父节点。递归定义顶点v的祖先（ancestor）为v和v的父节点的祖先（如果有的话）。

a）证明在算法B中的栈里如果v在u的下面，则v的父节点是u的一个祖先。

b）进一步，在B6步如果`COLOR`(v)=`COLOR`(u)，则v在当前栈中。

c）使用上述事实扩展算法B，当所给定的图不是二分图时，输出奇数长度的

回路中各顶点的名称。

73. [15] *random_graph*(10，45，0，0，0，0，0，0，0，0)的另一个名字是什么？

74. [21] *roget*(1022，0，0，0)的哪些顶点有最大的出度？

75. [22] SGB图生成器*board*(n_1，n_2，n_3，n_4，p，w，o)创建一个图，其顶点是t维整数向量$(x_1, \cdots, x_t)(0 \leqslant x_i < b_i)$，由前4个参数($n_1$，$n_2$，$n_3$，$n_4$)按如下方式所决定：令$n_5 \leftarrow 0$且$j \geqslant 0$是满足$n_{j+1} \leqslant 0$的最小的数。如果$j=0$，则令$b1 \leftarrow b2 \leftarrow 8$且$t \leftarrow 2$；这是默认的8×8的棋盘。否则如果$n_{j+1}=0$，则令$t=j$且对$1 \leqslant i \leqslant j$令$b_i \leftarrow n_i$。最后，如果$n_{j+1}<0$，则令$t \leftarrow |n_{j+1}|$，且令$b_i$为循环序列($n_1$，…，$n_j$，$n_1$，…，$n_j$，$n_1$，…)第$i$个元素。（例如，($n_1$，$n_2$，$n_3$，$n_4$)=(2，3，5，−7)是一组巧妙的参数，它产生的是一个7维的棋盘，其中$(b_1, \cdots, b_7) = (2, 3, 5, 2, 3, 5, 2)$，因此这个图有$2 \cdot 3 \cdot 5 \cdot 2 \cdot 3 \cdot 5 \cdot 2 = 1800$个顶点。）

剩下的参数(p，w，o)，表示“片（piece）、裹（wrap）和orientation（方向）”，决定了图中的弧。首先假设$w=o=0$，如果$p>0$，当且仅当$y_i=x_i+\delta_i(1 \leqslant i \leqslant t)$，我们有$(x_1, \cdots, x_t) \to (y_1, \cdots, y_t)$，这里$(\delta_1, \cdots, \delta_t)$是等式$\delta_1^2+\cdots+\delta_t^2=|p|$的一个整数解。如果$p<0$，我们允许$y_i=x_i+k\delta_i(k \geqslant 1)$，其对应于$k$次相同方向的移动。

如果$w \neq 0$，用二进制表示$w=(w_t \cdots w_1)_2$。则允许“返转”（wraparound），即对每个使$w_i=1$的i有$y_i=(x_i+\delta_i) \bmod b_i$或$y_i=(x_i+k\delta_i) \bmod b_i$。

如果$o \neq 0$，这是一个有向图；仅当偏移量$(\delta_1, \cdots, \delta_t)$按字典序大于$(0, \cdots, 0)$时才生成弧。但是当$o=0$时，图是无向的。

找出使board能产生下列基本图的(n_1，n_2，n_3，n_4，p，w，o)的值：(a)完全图K_n；(b)路径P_n；(c)回路C_n；(d)传递竞赛图（transitive tournament）$K_n^{\rightarrow}$；(e)有向路径$P_n^{\rightarrow}$；(f)有向回路$C_n^{\rightarrow}$；(g)$m \times n$的网格$P_m \square P_n$；(h)$m \times n$的圆筒$P_m \square C_n$；(i)$m \times n$的圆环面$C_m \square C_n$；(j)$m \times n$的行车路线图（rook graph）$K_m \square K_n$；(k)$m \times n$的有向圆环面$C_m^{\rightarrow} \square C_n^{\rightarrow}$；(l)零（null）图$\overline{K_n}$；(m)有$2^n$个顶点的$n$立方体$P_2 \square \cdots \square P_2$。

76. [20] *board*(n_1，n_2，n_3，n_4，p，w，o)能产生回路，或是平行（重复的）边吗？

77. [M20] 如果图G的直径不小于3，证明$\overline{G}$的直径不大于3。

78. [M26] 设$G=(V, E)$是一个图，其中$|V| = n$且$G \cong \overline{G}$。（换句话说，G是自补的：存在V中u—v间的一个置换φ，当且仅当$\varphi(u) \not- \varphi(v)$且$u \neq v$。我们可以想象$K_n$的边被涂成了黑色或白色；白色的边构成的图同构于黑色的边构成的图。）

a）证明$n \bmod 4$等于0或1。画出$n<8$的所有这种图。

b）证明如果$n \bmod 4$等于0，置换φ的每个回路的长度都是4的倍数。

c）相反地，每个带回路的置换φ出现在像G这种图中。

d）将结果扩展到$n \bmod 4$等于1的情况。

▶ 79. [M22] 给定$k \geqslant 0$，在顶点{0，1，…，4k}上构造一个规整且自补的图。

▶ 80. [M22] 根据习题77，一个自补图的直径必须为2或3。给定$k \geqslant 2$，构造具有这两个可能直径的自补图，(a)V={1，2，…，4k}；(b)V={0，1，2，…，4k}。

81. [20] 通过扩展(35)和(36)可定义一个简单无环有向图的补，使在$\overline{D}$中有$u \to v$当且仅当在D中$u \neq v$且$u \not\to v$。阶为3的自补有向图是什么？

82. [M21] 下列关于线图的描述是正确的还是错误的？

a）如果G被G'包含，则$L(G)$是$L(G')$的一个导出子图。

b）如果G是一个正则图，则$L(G)$也是。

c）对于所有m，$n>0$，$L(K_{m,n})$是正则的。

d）对于所有m，n，$r>0$，$L(K_{m,n,r})$是正则的。

e）$L(K_{m,n}) \cong K_m \square K_n$。

f）$L(K_4) \cong K_{2,2,2}$。

g）$L(P_{n+1}) \cong P_n$。

h）G和$L(G)$有同样数量的连通分量。

83. [16] 画出图$\overline{L(K_5)}$。

▶ 84. [M21] 图$L(K_{3,3})$是否是自补的？

85. [M22] (O. Ore，1962.) 对哪一个图我们有$G \cong L(G)$？

86. [M20] (R. J. Wilson.) 找出一个6阶图G满足$\overline{G} \cong L(G)$。

87. [20] Petersen图是否是(a)可3着色的？(b)边可3着色的？

88. [M20] 图$W_n=K_1 — C_{n-1}$被称作n阶的轮子，当$n \geqslant 4$时。它的子图中有多少个回路？

89. [M20] 证明结合律(42)和(43)。

▶ 90. [*M24*] 一个图被称作补图（cograph），如果它可以像代数运算一样使用求补或求和操作从单位图构造出。例如，有4个非同构的3阶图，它们都是补图：$\overline{K_3} = K_1 \oplus K_1 \oplus K_1$和它的补图$K_3$；$\overline{K_{1,3}} = K_1 \oplus K_2$和它的补图$K_{1,2}$，此处$K_2 = \overline{K_1 \oplus K_1}$。

穷尽枚举的结果表明只有11个非同构的4阶图。给出代数公式证明他们中的10个是余图。哪一个不是呢？

▶ 91. [20] 画出这些4顶点图(a)$K_2 \square K_2$；(b)$K_2 \otimes K_2$；(c)$K_2 \boxtimes K_2$；(d)$K_2 \triangle K_2$；(e) $K_2 \circ K_2$；(f) $\overline{K_2} \circ K_2$；(g) $K_2 \circ \overline{K_2}$。

92. [21] 文中定义的五种类型的图的乘积对简单的有向图和普通图都能很好地适用。画出如下4个顶点的有向图。

(a) $\vec{K_2} \square \vec{K_2}$；(b) $\vec{K_2} \otimes \vec{K_2}$；(c) $\vec{K_2} \boxtimes \vec{K_2}$；(d) $\vec{K_2} \triangle \vec{K_2}$；(e) $\vec{K_2} \circ \vec{K_2}$。

93. [15] 在这5种图的积中，哪一种使K_m和K_n的计算结果为K_{mn}？

94. [10] SGB中的*words*图是否是$P_{26} \square P_{26} \square P_{26} \square P_{26} \square P_{26}$的导出子图？

95. [M20] 如果G中顶点u的度为d_u且H中顶点v的度为d_v，在下面这些图中顶点(u, v)的度是多少？(a) $G \square H$；(b) $G \otimes H$；(c) $G \boxtimes H$；(d)$G \triangle H$；(e)$G \circ H$。

▶ 96. [M22] 设A是一个$m \times m'$的矩阵，其中$a_{uu'}$表示位于第u行、第u'列的元素；设B是一个$n \times n'$的矩阵，其中$b_{vv'}$表示位于第v行、第v'列的元素。直积$A \otimes B$是一个$mn \times m'n'$的矩阵，其中位于第(u, v)行、第(u', v')列的元素是$a_{uu'}b_{vv'}$。因此如果A和B分别是G和H的邻接矩阵，则$A \otimes B$是$G \otimes H$的邻接矩阵。

为这些图的邻接矩阵找出类似公式(a) $\boldsymbol{G} \square \boldsymbol{H}$；(b) $\boldsymbol{G} \boxtimes \boldsymbol{H}$；(c) $\boldsymbol{G} \triangle \boldsymbol{H}$；(d) $\boldsymbol{G} \circ \boldsymbol{H}$。

97. [*M*25] 找出图的和与积之间尽可能多的有趣的代数关系。(例如，对矩阵的直和与直积的分配律 $(\boldsymbol{A} \oplus \boldsymbol{B}) \otimes \boldsymbol{C} = (\boldsymbol{A} \otimes \boldsymbol{C}) \oplus (\boldsymbol{B} \otimes \boldsymbol{C})$ 可推出 $(G \oplus G') \otimes H = (G \otimes H) \oplus (G' \otimes H)$。我们也有 $\overline{K_m} \square H = H \oplus \cdots \oplus H$ (m个$\boldsymbol{H}$)等。)

98. [M20] 如果图G有k个连通分量，图H有l个连通分量，在图$G \square H$和$G \boxtimes H$中有多少个连通分量？

99. [M20] 设$d_G(u, u')$是图G中从顶点u到顶点u'的距离。证明$d_{G \square H}((u, v), (u', v')) = d_G(u, u') + d_H(v, v')$，且为$d_{G \boxtimes H}((u, v), (u', v'))$找到一个类似的公式。

100. [M21] 对哪些连通图G?H是连通的？

101. [M25] 找出所有连通图G和H，使$G \square H \cong G \otimes H$。

102. [*M*20] 在$m \times n$棋盘上，王的移动（king move）（横、竖或斜移一格）所形成的图的简单代数公式是什么？

103. [20] 完成表(54)，并对序列866444444应用算法H。

104. [18] 解释步骤H3和H4中对变量i、t和r的操作。

105. [M34] 假设$d_1 \geqslant \cdots \geqslant d_n \geqslant 0$，如在算法H中一样，设$c_1 \geqslant \cdots \geqslant c_{d_1}$为其共轭(conjugate)。证明$d_1 \cdots d_n$是一个图的度序列当且仅当$d_1 + \cdots + d_n$是偶数，并且$d_1 + \cdots + d_k \leqslant c_1 + \cdots + c_k - k$ $(1 \leqslant k \leqslant s)$，其中$s$是使$d_s \geqslant s$成立的最大值。

106. [20] 判断正误：若$d_1 = \cdots = d_n = d < n$且$nd$是偶数，则算法H生成一个连通图。

107. [*M*21] 证明一个自补图的度序列$d_1 \cdots d_n$满足$d_j + d_{n+1-j} = n-1$及$d_{2j-1} = d_{2j}$ $(1 \leqslant j \leqslant n/2)$。

▸ 108. [M23] 设计一个类似算法H的算法，生成顶点集为$\{1, \cdots, n\}$的简单有向图，使得当至少一个这样的图存在时，每个顶点k的入度和出度分别取特定值d_k^-和d_k^+。

109. [M20] 设计一个类似算法H的算法，生成一个顶点集为$\{1, \cdots, m+n\}$的二分图，尽可能使每个顶点k的度为指定值d_k；所有边j—k需满足$j \leqslant m$及$k > m$。

110. [M22]不使用算法H，通过直接构造说明当$n > d_1 \geqslant \cdots \geqslant d_n \geqslant d_1 - 1$且$d_1 + \cdots + d_n$是偶数时，序列$d_1 \cdots d_n$是图的度序列。

▸ 111. [25] 设G是顶点集为$V = \{1, \cdots, n\}$的图，k的度为d_k，并且$\max(d_1, \cdots, d_n) = d$。证明存在整数$N (n \leqslant N \leqslant 2n)$以及顶点集为$\{1, \cdots, N\}$的图$H$，使得$H$是度为$d$的正则图，且$H | V = G$。解释如何生成一个这样的正则图且使$N$尽可能小。

▸ 112. [20] 网络*miles* (128, 0, 0, 0, 0, 127, 0)中是否有三个等距的城市？如果不存在，哪三个城市可以组成最接近等边三角形的图形？

113. [05] 设H是一个m条边、n个顶点的超图，它的关联矩阵有多少行、多少列？

114. [M20] 假设多重图(26)被当成超图，关联矩阵是什么？相应的二分图呢？

▸ 115. [M20] 设$\boldsymbol{B}$是图G的关联矩阵，解释对称矩阵$\boldsymbol{B}^T\boldsymbol{B}$和$\boldsymbol{B}\boldsymbol{B}^T$的意义。

116. [M17] 描述完全二分 r—一致超图 $K_{m,n}^{(r)}$ 的边。

117. [M22] 互不同构的m条边、n个顶点的1—一致超图有多少？（边可以是重复的。）列举m=4和n=3时的所有情况。

118. [M20] “超森林（hyperforest）”是无回路的超图。若一个超森林有m条边、n个顶点、p个连通分量，所有点的度数总和是多少？

119. [M18] 对应(60)的超图去掉最后项（$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$）是怎样的图？

120. [M20] 对有向图进行推广以定义有向超图（directed hypegraph）。

121. [M19] 对于超图$H = (V, E)$，令$I(H) = (V, F)$，其中F是H所有的最大独立集所组成的集合。用$|V|$、$|F|$和$\alpha(I(H)^{T})$来表达$\chi(H)$。

▶ 122. [M24] 找出以下系统的最大独立集和最小着色数：(a)超图(56)；(b)Petersen图的对偶。

123. [17] 说明$K_n \square K_n$的最优着色问题等价于一个著名的组合问题的解。

124. [M22] Chvátal图（图2(f)）的着色数是多少？

125. [M48] 周长g取何值时存在一个4正则、4着色的图？

▶ 126. [M22] 找出当$m, n \geqslant 3$时“王移动圆环面”（kingwise torus）$C_m \boxtimes C_n$的最佳着色。

127. [M22] 当G是n阶图时，证明(a) $\chi(G)+\chi(\overline{G}) \leqslant n+1$；(b) $\chi(G)\chi(\overline{G}) \geqslant n$，并找出使等号成立的图。

128. [M18] 用$\chi(G)$和$\chi(H)$表达$\chi(G \square H)$，其中G和H都是图。

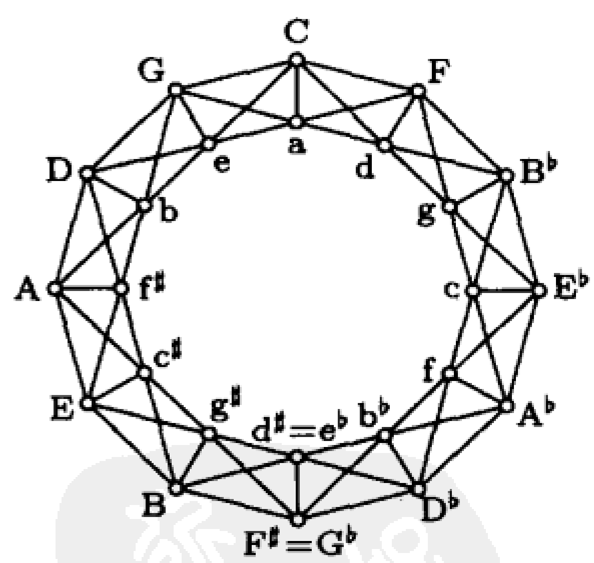
129. [23] 描述8×8的皇后图 (37)的最大团。

130. [M20] 在完全k分图中有多少最大团？

131. [M30] 设$N(n)$是n个顶点的图能有的最大团的最大数量。证明 $3^{\lfloor n/3 \rfloor} \leqslant N(n) \leqslant 3^{\lceil n/3 \rceil}$。

▶ 132. [M20] 若$\chi(G) = \omega(G)$，我们称G 可紧密着色（tightly colorable）。证明当G和H是可紧密着色时，有$\chi(G \boxtimes H) = \chi(G)\chi(H)$。

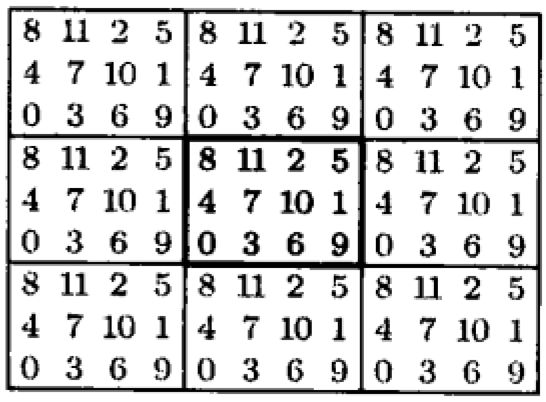
133. [21] 可以利用这里的“音乐图”（musical graph）复习本章中提到的各种定义，因为它的性质很容易被分析。确定它的(a)阶；(b)规模；(c)周长；(d)直径；(e)独立数$\alpha(G)$；(f)着色数$\chi(G)$；(g)边着色数$\chi(L(G))$；(h)团数$\omega(G)$；(i)著名的更小的图代数公式；(j)最小顶点覆盖的规模；(k)最大匹配；(l)G是否正则？(m)可画成平面图？(n)连通？(o)有向？(p)自由树？(q)哈密顿图？

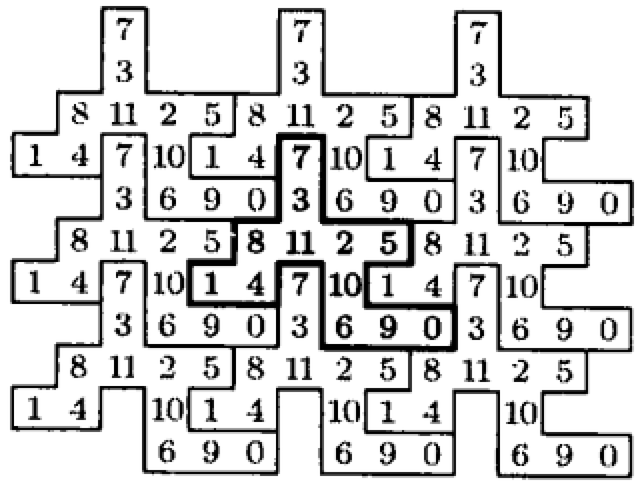


134. [M22] 音乐图有多少自同构？

▶ 135. [HM26] 假设作曲家在音乐图中随机选择了一条路，从顶点C出发共走了5步，每一步做等可能的选择。说明每走偶数步后，这条路走到顶点C的可能性比其他点更大。用12步从C走到C的精确概率是多少？

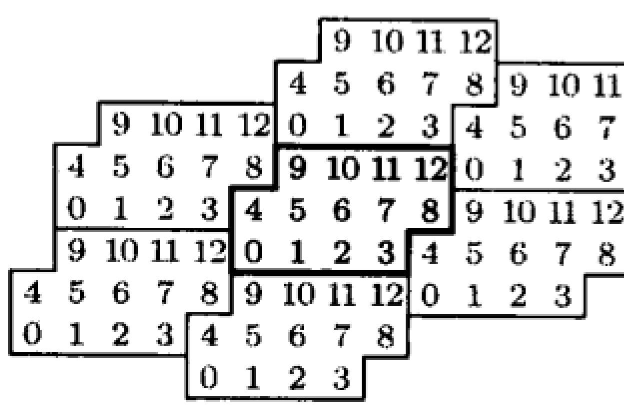
136. [HM23] Cayley有向图是顶点V为一个群的元素的有向图，它的弧是$v \to v\alpha_j$（对于$1 \leqslant j \leqslant d$及所有的顶点$v$），此处$(\alpha_1, \cdots, \alpha_d)$为群中不变的元素。Cayley图是将Cayley有向图作为图看待。Petersen图是否是Cayley图？





▸137. [M25] （广义圆环面（generalized torus））一个$m\times n$圆环面可以看成平面的平铺（tiling）。例如，我们可以想象无限个(50)中的3×4的环面珊格状地叠在一起，如上方左边的图所示；从每个顶点我们可以向东南西北的格点移动。这些顶点被标了号，从而向北的移动从v移到了$(v+4)$ mod 12，而向东的移动移到了$(v+3)$ mod 12等。上方右边的图展示了同一个圆环面，使用了不同形状的平铺方式；以任何方式选取标号为{0，1，⋯，11}的12个单元都会平铺这个平面，且使用相同底板图形（underlying graph）。

若某个形状形成广义圆环面，它的不同副本的切换也会平铺平面，其中单元(x, y)对应$(x+a, y+b)$和$(x+c, y+d)$中相同的顶点，(a, b)和(c, d)是整数向量并且$n=ad-bc>0$。从而这个广义圆环面有n个顶点。在上面的3×4的例子中，这些顶点(a, b)和(c, d)是$(4, 0)$和$(0, 3)$，当它们相对是$(5, 2)$和$(1, 3)$时，我们得到



这里$n=13$，并且向北的移动从v移到了$(v+4)$ mod 13；向东的移动移到了$(v+1)$ mod 13。

证明若$gcd(a, b, c, d)=1$，则这个广义圆环面的顶点总可以设置整数标号$\{0, 1, \cdots, n-1\}$，并且存在整数p和q，使得v的邻居是$(v\pm p)$ mod n 和$(v\pm q)$ mod n。

138. [HM27] 接习题137，有什么好的方法标记k维顶点$x=(x_1, \cdots, x_k)$，其中整数向量α_j满足每个顶点x等价于$x+\alpha_j$ $(1\leqslant j\leqslant k)$？在$k=3$，$\alpha_1=(3，1，1)$，$\alpha_2=(1，3，1)$，$\alpha_3=(1，1，3)$的情形下阐释你的方法。

▸139. [M22] 设H为阶为h的固定图，并且设$\#(H{:}G)$是H作为导出子图出现在G中的次数。若G是从所有$2^{n(n-1)/2}$个顶点集为$V=\{1, 2, \cdots, n\}$的图中随机选出的一个，求H是以下情况时$\#(H{:}G)$的平均值。

(a)K_h；(b)$P_h\,(h>1)$；(c) $C_h\,(h>2)$；(d)任意一个图。

140. [M30] G称为成比例的（proportional），若它的导出子图中

$\#(K_3{:}G)$、$\#(\overline{K_3}:G)$和$\#(P_3{:}G)$均满足习题139中期望的值。

a）说明习题88中的轮图（wheel graph）W_8是成比例的。

b）证明G是成比例的当且仅当 $\#(K_3:G)=\frac{1}{8}\binom{n}{3}$，并且顶点的度的序列$d_1\cdots d_n$满足等式

$$d_1+\cdots+d_n=\binom{n}{2}，\quad d_1^2+\cdots+d_n^2=\frac{n}{2}\binom{n}{2}。\qquad (*)$$

141. [26] 只有当$n \bmod 16\in\{0, 1, 8\}$时，习题140(b)的情况才能满足。编写程序寻找所有有n=8个顶点的成比例的图。

142. [M30]（S. Janson和J. Kratochvil，1991）证明有4或更多个顶点的图不是超比例的（extraproportional）的，即它的子图数$\#(H{:}G)$与习题139中对11个互不同构的H的阶为4的子图任一个的期望值一致。提示：$(n-3)\#(K_3{:}G)=4\#(K_4{:}G)+2\#(K_{1,1,2}{:}G)+\#(K_1\oplus K_3{:}G)$。

▸ 143. [M25] 设矩阵$\boldsymbol{A}$有不同的$m>1$行、$n\geqslant m$列。证明$\boldsymbol{A}$至少有一列可被删除并且可以保证每行仍不同。

▸ 144. [21] 设$\boldsymbol{X}$是$m\times n$矩阵，其中元素x_{ij}是0、1或*。$\boldsymbol{X}$的完备化是矩阵$\boldsymbol{X}^*$，其中每个*都被0或1代替。说明寻找有最少不同行的完备化问题等价于寻找图的着色数问题。

▸ 145. [25]（R. S. Boyer和J. S. Moore，1980）假设向量$a_1\cdots a_n$包含一个多数元素，即这个值出现超过$n/2$次。设计一个算法在少于n次比较后找到它。提示：若$n\geqslant 3$且$a_{n-1}\neq a_n$，$a_1\cdots a_n$的主要元素也是$a_1\cdots a_{n-2}$的主要元素。

你们这些智慧之士依旧得屈尊纡贵。

——乔纳森斯·威夫特（Jonathan Swift），《Cadenus and Vanessa》（1713）

如果以2为基，则结果可称为二进制数字，

或使用由J. W. Tukey所建议的更简短的单词——比特。

——Claude E. Shannon，《Bell System Technical Journal》（1948）

bit（钻头），名词，一种打孔工具……

——《蓝登书屋英语大词典》（1987）

7.1 0和1

组合算法通常需要特别关注效率，合适的数据表示是获得必要速度的重要途径。因此在详细研究组合算法之前补充一下基本数据表示技术是明智的。

今天大多数的计算机都是基于二进制数字系统，而不是直接工作于人类所喜爱的十进制，这是因为机器特别善于处理通常被标记为0和1的开和关这两个状态。但从第1～6章，对于一些问题，我们没有充分利用这样一个事实：在某些方面二进制计算机能做得更快，而十进制计算机却无法做到。二进制计算机通常执行"逻辑"或"按位"计算就像其执行加减法一样容易，而我们却很少充分利用该能力。我们已经看到二进制和十进制计算机并非截然不同，但对于某些场合，我们有时会在一定程度上要求二进制计算机能更容易、更轻松地执行某些操作。

使用0和1可对信息进行编码、对信息间的逻辑关系进行编码，甚至对处理信息的算法进行编码，这一令人诧异的能力使对二进制数字进行研究所得的成果特别丰富。事实上，我们不仅可使用按位操作以加强组合算法，我们也发现二进制逻辑的一些性质也很自然地导致了一些新的组合问题，它们本身也很有意思。

在驯服宇宙中杂乱无序的0、1并使其完成有益工作的过程之中，计算机科学家变得越来越成熟。但作为世界舞台上的比特玩家，在开始研究高层次的概念和技术之前，我们最好对二进制量的底层属性有一个完全的了解。下面开始研究对单一比特和比特序列进行组合的基本方式。

7.1.1 布尔基础

将两个给定的比特x和y转换为第3个比特$z=f(x, y)$的可能函数共有16种，因为对$f(0, 0)$、$f(0, 1)$、$f(1, 0)$、$f(1, 1)$中的任何一个都有2种选择。表1是形式逻辑研究中与这些函数相联系的名字和记法，这里假定1对应于"真"，0对应于"假"。$f(0, 0)$、$f(0, 1)$、$f(1, 0)$和$f(1, 1)$这4个值的序列习惯上称为函数f的真值表（truth table）。

让我们构想一个代数系统，其中符号x，y，z&c可以没有差别地取值0和1，或者就是这些值中的一个。

——乔治·布尔（Geogre Boole），《An Investigation of the Laws of Thought》（1854）

特维德里迪（Tweed ledee）继续说道："反之，如果它是如此，它就可能如此；如果它万一如此，它也将是如此；但如果它不是如此，那它肯定不是。这就是逻辑。"

——刘易斯·卡洛尔（Lewis Carroll），《Through the Looking Glass》（1871）

为了纪念乔治·布尔，这些函数通常被称为"布尔操作"，布尔首先发现了0和1上的代数操作可用于逻辑系统中推理演算的构造。[《The Mathematical Analysis of Logic》，(Cambridge：1847)；《An Investigation of the Laws of Thought》(London：1854)。]但布尔事实上从来没有处理过"逻辑或"操作∨，他将自己严格局限于通常的0、1之上的算术运算。因此，他将析取表示为$x+y$，但他忍痛不使用该记法，除非x和y是互斥的（即不皆为1）。如果必要，他将其写为$x+(1-x)y$以确保析取的结果不会等于2。

在将+操作用英语翻译时，布尔有时称其为"and"，有时又称其为"or"。在当代数学家看来这似乎有点奇怪，但事实上，我们要认识到布尔使用的是英语中的惯用法，例如，我们常说"男孩和女孩都是孩子"，但不会说"孩子是男孩或女孩"。

W. Stanley Jevons [《Pure Logic》(London：Edward Stanford，1864)，§69]对布尔演算进行了扩充，其包括了一条非常规规则$x+x=x$，他指出基于新的+操作，我们有$(x+y)z=xz+yz$。但Jevons不知道另一条分配律$xy+z=(x+z)(y+z)$的存在。据推测，他之所以意识不到，是和其使用的记法有关，因为第二条分配律在算术上没有熟悉的对应物，表1使用了更加对称的记法$x\wedge y$、$x\vee y$，这使我们能更容易记住这两条分配律。

$$(x\vee y)\wedge z=(x\wedge z)\vee(y\wedge z) \tag{1}$$

$$(x\wedge y)\vee z=(x\vee z)\wedge(y\vee z) \tag{2}$$

第二条定律由C. S. Peirce引入，Peirce独立发现了对布尔演算进行扩充的方式[《Proc. Amer. Acad. Arts and Sciences》7 (1867)，250－261]。顺便提一句，在对这些早期进展进行讨论的数年之后，Peirce仍将其称为"带Jevons加法的布尔代数"，其实那个时候他所不熟悉的"Boolean"这个词已经被使用了多年，但在1963年这个单词才出现在Funk与Wagnalls编著的非缩略语字典中。

真值组合记法的出现要比布尔代数早得多。事实上，命题逻辑早在公元前4世纪就由希腊哲学家发展起来了。那时就有关于如何对命题"如果x则y"进行合适的真假赋值的争论，这里x和y都是命题；大约公元前300年，麦加的斐洛（Philo）就借助于如表1所示的真值表给出了上述命题的定义，特别是，斐洛说明了当x和y

都是假时，隐含式的值为真。这些早期工作的大部分已经遗失了，但在伽林（Galen）的（公元2世纪）著作中有若干章节涉及命题的可兼或和不可兼或的讨论。[参阅I. M. Bocheński著的《Formale Logik》（1956）和由Ivo Thomas（1961）翻译的英文版，书中包括了从远古时代到20世纪的逻辑发展的极好概览。]

表1 16个两元逻辑操作

真值表	记法	操作符∘	名字
0000	0	$\perp$	矛盾；假；自相矛盾；常量0
0001	$xy, x\wedge y, x\&y$	$\wedge$	合取；并且
0010	$x\wedge\bar{y}$, $x\not\supset y\ [x>y]$, $x\dot{-}y$	$\supseteq$	非隐含；不同；但非
0011	x	L	左投影
0100	$\bar{x}\wedge y$, $x\not\subset y$, $[x<y]$, $y\dot{-}x$	$\overline{\subset}$	反向非隐含；不是……而是
0101	y	R	右投影
0110	$x\oplus y$, $x\not\equiv y$, $x\wedge y$	$\oplus$	异或；非等价；“xor”
0111	$x\vee y$, $x\mid y$	$\vee$	（可兼）析取；或；与/或
1000	$\bar{x}\wedge\bar{y}$, $\overline{x\vee y}$, $x\,\bar{\vee}\,y$, $x\downarrow y$	$\bar{\vee}$	或非；联合否定；既不……也不……
1001	$x\equiv y$, $x\leftrightarrow y$, $x\Leftrightarrow y$	$\equiv$	等价；当且仅当；“iff”
1010	$\bar{y}$, $\neg y$, $!y$, $\sim y$	$\bar{\mathrm{R}}$	右补
1011	$x\vee\bar{y}$, $x\subset y$, $x\Leftarrow y$, $[x\geqslant y]$, x^{y}	$\subset$	拟隐含；如果
1100	$\bar{x}$, $\neg x$, $!x$, $\sim x$	$\bar{\mathrm{L}}$	左补
1101	$\bar{x}\vee y$, $x\supset y$, $x\Rightarrow y$, $[x\leqslant y]$, y^{x}	$\supset$	隐含；仅当；如果……则……
1110	$\bar{x}\vee\bar{y}$, $\overline{x\wedge y}$, $x\,\bar{\wedge}\,y$, $x\mid y$	$\bar{\wedge}$	与非；非既……又……；“nand”
1111	1	$\top$	事实的陈述；有效；重言式；常量1

一个有两个变量的函数通常记为$x\circ y$而非$f(x, y)$，这里∘是某个合适的操作符。表1说明了对于有两个变元的布尔函数所采用的16种操作符，例如，$\perp$是真值表为0000的函数的符号化表示，$\wedge$对应真值表中0001的符号，$\supseteq$对应0010的符号等。我们有$x\perp y=0$，$x\wedge y=xy$，$x\supseteq y=x\dot{-}y$，$x\,\mathrm{L}\,y=x$，$\cdots$，$x\,\bar{\wedge}\,y=\bar{x}\vee\bar{y}$，$x\top y=1$。

当然，表1中每个操作的重要性并不是一样的。例如，第一个和最后一个操作由于其值是独立于x和y的常量，因此都是不重要的。其中有4个只是单独的x或y的函数。我们将$1-x$记为$\bar{x}$，它是x的补（complement）。

真值表仅包含单独一个1的4种操作可方便地表示为AND操作$\wedge$，即$x\wedge y$，$x\wedge\bar{y}$，$\bar{x}\wedge y$，$\bar{x}\wedge\bar{y}$。有3个1的则可方便地表示为OR操作$\vee$，即$x\vee y$，$x\vee\bar{y}$，$\bar{x}\vee y$和$\bar{x}\vee\bar{y}$。已经证明基本函数$x\wedge y$和$x\vee y$比其补或半补函数更加有用，而NOR和NAND操作$x\,\bar{\vee}\,y=\bar{x}\wedge\bar{y}$和$x\,\bar{\wedge}\,y=\bar{x}\vee\bar{y}$，由于使用晶体管电路易于实现，也很有意义。

1913年，H. M. Sheffer说明了所有16种函数皆能用一种函数予以表示，它们是$\bar{\vee}$或$\bar{\wedge}$。事实上，C. S. Peirce大约在1880年就已发现相同的结果，但他的作品直到他去世之后才发表。[《Collected Papers of Charles Sanders Peirce》4（1933），§§12—20，264]。表1显示了NAND和NOR有时被写成$x|y$和$x\downarrow y$，有时被称为“Sheffer竖划线”或“Peirce箭头”。今天，最好不要使用Sheffer竖划线来表示NAND，因为$x|y$在如C这样的程序设计语言中表示按位或操作，即$x\vee y$。

到目前为止，除了两个函数之外，我们已经讨论了表1中的所有函数。剩下的两个函数是$x\equiv y$和$x\oplus y$，即“等价”和“不可兼或”（异或），它们之间的关系由下面的等式刻画。

$$x\equiv y=\bar{x}\oplus y=x\oplus\bar{y}=1\oplus x\oplus y \tag{3}$$

$$x\oplus y=\bar{x}\equiv y=x\equiv\bar{y}=0\equiv x\equiv y \tag{4}$$

这两个操作都满足结合律（参见习题6）。在命题逻辑中，等价比异或更为重要，异或指的是不等价，但如果考虑计算机字上的按位操作，情况则恰恰相反，如7.1.3节所见，在编写一般程序时，异或比等价更有用。甚至在仅仅涉及一个位的运算时，$x\oplus y$也有其重要的应用，主要原因是存在如下事实。

$$x\oplus y=(x+y) \bmod 2 \tag{5}$$

因此在由两个元素构成的域中，$x\oplus y$和$x\wedge y$可分别表示加法和乘法（参见4.6节），因此$x\oplus y$自然就继承了一些“纯粹”的数学性质。

基本等式 现在来看一下基本操作符$\wedge$、$\vee$、$\oplus$和$^-$之间的交互作用，其他操作容易借助这4个操作符加以表示。$\wedge$、$\vee$和$\oplus$中的每一个都是可结合、可交换的。除了分配律(1)和(2)外，我们还有

$$(x\oplus y)\wedge x=(x\wedge y)\oplus(y\wedge z) \tag{6}$$

和吸收律

$$(x\wedge y)\vee x=(x\vee y)\wedge x=x \tag{7}$$

最简单且重要的等式是

$$x\oplus x=0 \tag{8}$$

应用显而易见的事实$x\oplus 0=x$，上述等式还蕴涵

$$(x\oplus y)\oplus x=y,\ (x\oplus y)\oplus y=x \tag{9}$$

换言之，给定$x\oplus y$以及x与y中的一个值，就可以推出另一个的值。另外，我们还不能看轻互补律（complementation law）

$$\bar{x}=x\oplus 1 \tag{10}$$

另一对重要的等式是以Augustus De Morgan的名字命名的De Morgan律，其说明“一个聚集物的逆是其聚集成分逆的组合，一个组合物的逆是其组成成分逆的聚集。因此（A，B）和AB的逆分别是ab和（a，b）。”[《Trans. Cambridge Philos. Soc.》10(1858), 208]。由表1中的NAND和NOR操作相联系的真值表可隐式导出这些规则的现代记法，即

$$\overline{x\wedge y}=\bar{x}\vee\bar{y} \tag{11}$$

$$\overline{x\vee y}=\bar{x}\wedge\bar{y} \tag{12}$$

顺便提一下，W. S. Jevons知道式（12）但不了解式（11），他将AB的补写成值一致的$\overline{A}B+\overline{B}A+\overline{A}\,\overline{B}$，而非$\overline{A}+\overline{B}$。然而，De Morgan并非是首位确切说明上述定

律的英国人。式（11）和（12）都可以在14世纪两位学院派哲学家的著作中找到：英国奥卡姆的William[《Summa Logicæ》2(1323)]和Walter Burley[《De Puritate Artis Logicæ》(c. 1330)]。

De Morgan定律和其他一些定律可用来表示∧、∨和⊕间的相互关系。

$$x \wedge y = \overline{\bar{x} \vee \bar{y}} = x \oplus y \oplus (x \vee y) \tag{13}$$

$$x \vee y = \overline{\bar{x} \wedge \bar{y}} = x \oplus y \oplus (x \wedge y) \tag{14}$$

$$x \oplus y = (x \vee y) \wedge \overline{x \wedge y} = (x \wedge \bar{y}) \vee (\bar{x} \wedge y) \tag{15}$$

依照7.1.2节的习题77，将表达式$x_1 \oplus x_2 \oplus \cdots \oplus x_n$转化为仅含操作∧、∨和¯的等价表达式所需的计算步骤至少为$4(n-1)$，因此，这3个操作符并非⊕的特别合适的替代者。

***n*个变元的函数** 一个有着3个布尔变元x、y和z的布尔函数$f(x, y, z)$可由8比特的真值表$f(0,0,0)\,f(0,0,1)\cdots f(1,1,1)$确定；一般地，每个$n$元布尔函数$f(x_1,\cdots,x_n)$对应一个$2^n$个比特的真值表，它按序列出了$f(0,\cdots,0,0)\ f(0,\cdots,0,1)\ f(0,\cdots,1,0),\cdots,f(1,\cdots,1,1)$的值。

我们无需为所有这些函数设计特别的名字和记法，所有这些函数都可以借助于前面已经学习过的二元函数予以表示。例如，如I. I. Zhegalkin所观察到的那样[《Matematicheskiĭ Sbornik》35(1928)，311—369]，对于$n>0$，总有合适的函数g和h使

$$f(x_1,\cdots,x_n) = g(x_1,\cdots,x_{n-1}) \oplus h(x_1,\cdots,x_{n-1}) \wedge x_n \tag{16}$$

这里可令

$$\begin{aligned} g(x_1,\cdots,x_{n-1}) &= f(x_1,\cdots,x_{n-1},0) \\ h(x_1,\cdots,x_{n-1}) &= f(x_1,\cdots,x_{n-1},0) \oplus f(x_1,\cdots,x_{n-1},1) \end{aligned} \tag{17}$$

按习惯，操作符∧的优先级比⊕高，因此式（16）的右边就不需要将子公式‘$h(x_1,\cdots,x_{n-1}) \wedge x_n$’放在括号里边了。对于$g$和$h$重复递归调用上述过程直到0元函数，此时得到的是一个仅包含操作符⊕、∧以及2^n个常量的序列。进一步，这些常量还可以使用下述等式简化。

$$x \wedge 0 = 0 \quad 且 \quad x \wedge 1 = x \oplus 0 = x \tag{18}$$

在应用结合律和分配律之后，可以看到，仅当$f(x_1,\cdots,x_n)=0$时，过程结束于常量0；仅当$f(0,\cdots,0)=1$时，过程结束于常量1。

例如

$$\begin{aligned} f(x,y,z) &= ((1 \oplus 0 \wedge x) \oplus (0 \oplus 1 \wedge x) \wedge y) \oplus ((0 \oplus 1 \wedge x) \oplus (1 \oplus 1 \wedge x) \wedge y) \wedge z \\ &= (1 \oplus x \wedge y) \oplus (x \oplus y \oplus x \wedge y) \wedge z \\ &= 1 \oplus x \wedge y \oplus x \wedge z \oplus y \wedge z \oplus x \wedge y \wedge z \end{aligned}$$

按照规则（5）和等式$x \wedge y = xy$，上述可简化为多项式

$$f(x, y, z) = (1 + xy + xz + yz + xyz) \bmod 2 \tag{19}$$

注意，对于每个变元上述多项式都是线性的（阶小于等于1）。一般地，通过类似计算，对任何一个布尔函数$f(x_1, \cdots, x_n)$，我们都可以得到与上述多项式类似的唯一表达式，一般称它为多重线性表示（multilinear representation），它是2^n的0个或多个可能项$1, x_1, x_2, x_1x_2, x_3, x_1x_3, x_2x_3, x_1x_2x_3, \cdots, x_1x_2\cdots x_n$的和（模2）。

布尔使用了一种不同的布尔函数分解法，对于实际中出现的函数种类，这种方法所得的结果更为简单。对应于（16），他将其写成

$$f(x_1,\cdots,x_n)=(g(x_1,\cdots,x_{n-1})\wedge\bar{x}_n)\vee(h(x_1,\cdots,x_{n-1})\wedge x_n) \qquad (20)$$

布尔称其为“展开律”（development law），由此（17）对应着下述简单形式

$$\begin{aligned} g(x_1,\cdots,x_{n-1})&=f(x_1,\cdots,x_{n-1},0)\\ h(x_1,\cdots,x_{n-1})&=f(x_1,\cdots,x_{n-1},1)\end{aligned} \qquad (21)$$

使用分配律(1)，消去常数，并重复迭代应用布尔规则，最后得到的公式是0个或多个极小项（minterm）的析取，这里极小项是如同$x_1\wedge\bar{x}_2\wedge\bar{x}_3\wedge x_4\wedge x_5$这样的表达式，其中每个变元或其补在其中出现。注意，每个极小项都是一个恰在一个点上为真的布尔函数。

例如，随机考虑一个函数$f(w, x, y, z)$，其真值表是

$$1100\ 1001\ 0000\ 1111 \qquad (22)$$

通过重复使用布尔规则（20）展开，可得一个由8个极小项的析取组成的函数，其中每一个极小项对应真值表中的一个1。

$$\begin{aligned} f(w,x,y,z)=&(\bar{w}\wedge\bar{x}\wedge\bar{y}\wedge\bar{z})\vee(\bar{w}\wedge\bar{x}\wedge\bar{y}\wedge z)\vee(\bar{w}\wedge x\wedge\bar{y}\wedge\bar{z})\vee(\bar{w}\wedge x\wedge y\wedge z)\\ &\vee(w\wedge x\wedge\bar{y}\wedge\bar{z})\vee(w\wedge x\wedge\bar{y}\wedge z)\vee(w\wedge x\wedge y\wedge\bar{z})\vee(w\wedge x\wedge y\wedge z)\end{aligned} \qquad (23)$$

一般地，极小项的析取称为满析取范式（full disjunctive normal form）。每个布尔函数都可以以这种方式表示，而最后的结果也是唯一的，当然这里不考虑极小项间的顺序。吹毛求疵地说，当$f(x_1, \cdots, x_n)$恒等于0时情况比较特殊。基于1.2.3节定义$\sum_{k=1}^{0}a_k=0$和$\prod_{k=1}^{0}a_k=1$同样的理由，可将0看成是不含任何项的空析取，1看成是不含任何项的空合取。

在《Amer. J. Math.》3 (1880), 37—39中，C. S. Peirce指出每一个布尔函数也有一个满合取范式（full conjunctive normal form），它是形如$\bar{x}_1\vee x_2\vee\bar{x}_3\vee\bar{x}_4\vee x_5$的最小子句（minclause）的合取。一个最小子句最多在一点为0，因此合取中的每一最小子句对应了真值表中的每个0。例如，(22)和(23)中的函数的满合取范式为

$$\begin{aligned} f(w,x,y,z)=&(w\vee x\vee\bar{y}\vee z)\wedge(w\vee x\vee\bar{y}\vee\bar{z})\wedge(w\vee\bar{x}\vee y\vee\bar{z})\wedge(w\vee\bar{x}\vee\bar{y}\vee z)\\ &\wedge(\bar{w}\vee x\vee y\vee z)\wedge(\bar{w}\vee x\vee y\vee\bar{z})\wedge(\bar{w}\vee x\vee\bar{y}\vee z)\wedge(\bar{w}\vee x\vee\bar{y}\vee\bar{z})\end{aligned} \qquad (24)$$

然而，无需诧异，我们平常涉及的析取和合取并非关于最小项和最小子句必然是满的。因此，按照Paul Bernays 在《Habilitationsschrift》（1918）中所引入的命名法，任意合取式的析取一般称为析取范式（disjunctive normal form）或“DNF”。

$$\bigvee_{j=1}^{m}\bigwedge_{k=1}^{s_j} u_{jk} = (u_{11}\wedge\cdots\wedge u_{1s_1})\vee\cdots\vee(u_{m1}\wedge\cdots\wedge u_{ms_m}) \tag{25}$$

这里u_{jk}是文字，即变量x_i或其补。类似地，可称任意文字析取的合取为合取范式（conjunctive normal form），或简称为“CNF”。

$$\bigwedge_{j=1}^{m}\bigvee_{k=1}^{s_j} u_{jk} = (u_{11}\vee\cdots\vee u_{1s_1})\wedge\cdots\wedge(u_{m1}\vee\cdots\vee u_{ms_m}) \tag{26}$$

现今大量嵌入在计算机芯片内部的电子线路是由“可编程逻辑阵列”（PLA）组成的，PLA对输入信号以及可能的补相与（AND）后再相或（OR）。换句话说，基本上PLA计算的是一个或多个析取范式。这种类型的构件速度快，用途多，价格廉。事实上，DNF自从20世纪50年代开始就在电子工程领域扮演着重要的角色，那时开关电路还由相对老式的设备如继电器和真空管实现。可以说，人们长期对寻找可表示布尔函数的最简单的DNF倍感兴趣。随着技术的持续发展，我们仍然可以认为对析取范式的进一步了解还是重要的事情。

由于析取式中的任何一个项为真隐含整个公式为真，因此DNF中的项通常也称为蕴涵项（implicant）。例如，在公式

$$f(x,y,z)=(x\wedge\bar{y}\wedge z)\vee(y\wedge z)\vee(\bar{x}\wedge y\wedge\bar{z})$$

中，我们知道当$x\wedge\bar{y}\wedge z$为真时，即$(x, y, z)=(1, 0, 1)$时，f 为真。但要注意的是，在本例中，尽管没有显式标出，较短的项$x\wedge z$也是f的蕴涵项，因为当$x=z=1$时，不管y的值为何，由于另一个项$y\wedge z$的存在，f的值必是真的。类似地，$\bar{x}\wedge y$也是该函数的蕴涵项。由此，我们也许应该使用更简单的公式

$$f(x,y,z)=(x\wedge z)\vee(y\wedge z)\vee(\bar{x}\wedge y) \tag{27}$$

此时，无法在蕴涵项中进行更多的删除了，因为不管是x、y、z或是$\bar{x}$都无法强大到足以蕴涵f的地步。

一个蕴涵项如果无法通过删除其中的文字使其弱化而进一步分解的话，该蕴涵项就称为基本蕴涵项（prime implicant），该术语由W. V. Quine在《AMM》59 (1952)，521－531中首先使用。

如果对记法进行简化并采用更加几何化的视图，理解这些基本概念就非常容易了。将x看成向量或看成长为n的二进制串，我们就可简单地把$f(x_1, \cdots, x_n)$ 记成$f(x)$。例如使（22）中的函数为真的串$wxyz$为

$$\{0000,0001,0100,0111,1100,1101,1110,1111\} \tag{28}$$

我们可以把它们想象为4维立方体$2\times2\times2\times2$上的8个点。（28）中的8个点对应了在满析取范式（23）中显式出现的极小蕴涵项；但其中没有一个是基本蕴涵项。如同在6.5节中讨论数据库查询那样，若用*表示“百搭牌”，则（28）中的前2点构成了子立方体000*，而后4点构成了子立方体11**；由此，$\bar{w}\wedge\bar{x}\wedge\bar{y}$是$f$ 的一个蕴涵项，而$w\wedge x$也是。类似地，子立方体0*00对应了（28）中的两个点，使$\bar{w}\wedge\bar{y}\wedge\bar{z}$ 成为f 的一个蕴涵项。

一般地，在这种方式下，每个基本蕴涵项对应一个最大子立方体，其位于使f为真的点集的内部。（一个子立方体是最大的，指其不位于任何一个具备相同性质的较大的立方体的内部，我们无法将任何一个显式比特位替换为*。一个最大子立方体有着数目最多的星号，因此有着数目最少的被包含的坐标点，也就是说其对应的蕴涵项有着最少数目的变量。）（28）中8个点的最大子立方体是

$$000*, 0*00, *100, *111, 11** \tag{29}$$

因此，（23）中的函数$f(w, x, y, z)$的基本蕴涵项是

$$(\bar{w} \wedge \bar{x} \wedge \bar{y}) \vee (\bar{w} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge z) \vee (w \wedge x) \tag{30}$$

一个布尔函数的析取基本型（disjunctive prime form）是其所有的基本蕴涵项的析取。给定使函数为真的点的列表，习题30包含了一个寻找给定函数的所有基本蕴涵项的算法。

以完全类似的方式，可以定义基本子句（prime clause）。基本子句是被f隐含的析取子句，其不含具有相同性质的子子句。而f的合取基本型（conjunctive prime form）是其所有基本子句的合取（习题19是其中的一个例子）。

对于许多简单的情景，析取基本型是一个函数能具有的最短的可能析取型。但通常还可以做得更好，因为我们可以使用少数几个极大子立方体就可以覆盖掉所有必要的点。例如，基本蕴涵项$y \wedge z$在（27）中并非必要，而在表达式（30）中，我们也不需要$(\bar{w} \wedge \bar{y} \wedge \bar{z})$和$(x \wedge \bar{y} \wedge \bar{z})$；在其他项出现时，它们没有一个是充分的。

遗憾的是，我们将在7.9节看到寻找最好析取范式的任务是NP完全的，因此通常来说这是一个很困难的任务。但对于比较小的问题，已经发明了许多有用的技巧，这些技巧的解释可参见E. J. McCluskey（New York：McGraw-Hill，1965）所著的《Introduction to the Theory of Switching Circuits》，后期的发展可见Petr Fišer和Jan Hlavička在《Computing and Informatics》22(2003), 19—51中的讨论。

然而，存在一个重要的最短DNF容易被刻画的特例。若一个布尔函数当变量的值从0变为1时，其值不会从1变为0，则称该函数是单调的（monotone）或正的（positive）。换言之，f是单调的当且仅当$x \subseteq y$时$f(x) \leqslant f(y)$。这里，当且仅当对所有j，$x_j \leqslant y_j$，则认为位串$x=x_1 \cdots x_n$是包含于或等于位串$y=y_1 \cdots y_n$的。一个等价的条件是（参见习题21）函数f或者是常量，或者可以完全不借助补运算而仅使用$\wedge$和$\vee$表示。

定理Q 一个单调布尔函数的最短析取范式是其析取基本型。

证明：[W. V. Quine，《Boletín de la Sociedad Matemática Mexicana》10 (1953)，64－70。]令$f(x_1, \cdots x_n)$是单调的且$u_1 \wedge \cdots \wedge u_s$是它的一个基本蕴涵项，显然$u_1 = \bar{x}_i$是不可能的，因为若如此，由单调性可知，更短的项$u_2 \wedge \cdots \wedge u_s$也将是一个蕴涵项。因此，基本蕴涵式不含补文字。

现设$u_1 \leftarrow \cdots \leftarrow u_s \leftarrow 1$，而其他变量为0，此时$f$的值将为1，但$f$的所有其他基本蕴涵项都消失了。因为一个最短DNF的每个蕴涵项必然是基本的，因此$u_1 \wedge \cdots \wedge u_s$

必在每一个最短的DNF之中。▌

推论Q 一个析取范式是一个单调布尔函数的析取基本型，当且仅当其不含补文字且其任何一个蕴涵项都不包含在其他蕴涵项之中。▌

可满足性 一个布尔函数如果不恒等于0，即至少有一个蕴涵项，则称其是可满足的。计算机科学中最著名的未解决问题之一，就是寻找一种有效的方式判定一个给定的布尔函数是否是可满足的。更精确地说，就是存在一个测试算法，当输入长为N的布尔公式时，在最多执行$N^{O(1)}$步之后，总是能给出正确的答案吗？

如果你是第一次听到这个问题，你可能会情不自禁地问自己："什么啊？你不会开玩笑吧。对如此简单的问题，计算机科学家会仍然没有答案？"

好吧，如果你认为可满足性测试是一件微不足道的事情，那就请告诉大家你的方法吧。我同意，问题并非都是困难重重的，例如，给定的公式仅仅涉及30个布尔变量，使用暴力测试法检查2^{30}——大约是10亿个实例，事实上就可以解决这个问题。由于数理逻辑是一种强有力的概念表示法，许多还在等待答案的实际问题也可使用布尔函数进行形式化描述，但也许会涉及100多个变量。求解这些问题就对应于寻找满足 $f(x)=1$的向量$x=x_1\cdots x_{100}$。因此，得到一个真正有效的可满足性问题的求解方法是一个巨大的成就。

至少有一种可满足性测试是不用动脑筋的：如果函数 $f(x_1, \cdots, x_n)$是随机选择的，则所有2^n种真值表项的可能都是均等的，则f基本上肯定是可满足的，在（平均）小于2次测试之后，就能找到使$f(x)=1$的x了。这就像反复扔投一个硬币直到正面朝上为止，并不需要等待很长时间。但隐情是，对于现实问题真值表中真值的分布并不是随机的。

好，我们承认，一般来说，满足性测试确实看起来是困难的。事实上，甚至在要求布尔函数为"3CNF公式"，即最多是由3个文字组成的子句的合取范式时，可满足性测试也是一个难的问题。

$$f(x_1,\cdots,x_n)=(t_1\vee u_1\vee v_1)\wedge(t_2\vee u_2\vee v_2)\wedge\cdots\wedge(t_m\vee u_m\vee v_m) \tag{31}$$

这里每个t_j、u_j和v_j是对于某个k的x_k或$\bar{x}_k$。确定3CNF表示的公式是否可满足的问题称为"3SAT"，习题39解释了为何它不比一般的可满足性问题容易。

我们将看到3SAT这一难解问题的许多例子，特别是在7.9节，在那里我们将详细讨论满足性测试问题。然而，问题有点特别，在考虑可满足性时，一个足够长的公式我们才需要思考两次。例如3CNF中最短的不可满足公式是$(x\vee x\vee x)\wedge(\bar{x}\vee\bar{x}\vee\bar{x})$，对于一个受过教育的人来说这并非一个挑战。除非一个子句的3个文字t_j、u_j和v_j对应3个不同的变量，我们是不会去趟这个浑水的。在这种情况下，每个子句恰好排除了1/8的可能性，因为（t_j，u_j，v_j）的7种不同设置都会使其为真。结果是，每一种此类的3CNF最多有7个子句是自动可满足的，其变量的随机设置将使成功的概率≥1−7/8=1/8。

因此最短的有意义的3CNF公式至少有8个子句。而事实上，基于6.5节的公式（13）中由R. L. Rivest设计的结合块（associative block），一个有意义的包含8个子句的公式确实是存在的。

$$(x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \\ \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \quad (32)$$

这8个子句中的任意7个都是可满足的，且恰好有2种可满足的方式，它们都影响了3个变量的取值，例如，前7个子句隐含着$x_1x_2x_3$=001。但8个子句的全集是不可能同时被满足的。

简单的特别情景 已经确认有两种重要的布尔公式类其可满足性问题是特别简单的。这两种情况就是被测试的合取范式仅包含“Horn子句”或“Krom子句”。Horn子句是文字的OR，其中所有或近乎所有的文字都是补文字，即至多只有一个文字是纯的，是上面没有横划线的变量。一个Krom子句是恰含2个文字的OR。例如

$$\bar{x} \vee \bar{y}, \quad w \vee \bar{y} \vee \bar{z}, \quad \bar{u} \vee \bar{v} \vee \bar{w} \vee \bar{x} \vee \bar{y} \vee z \text{ 和} x$$

是Horn子句，而

$$x \vee x, \quad \bar{x} \vee \bar{x}, \quad \bar{x} \vee \bar{y}, \quad x \vee \bar{y}, \quad \bar{x} \vee y \quad \text{和} x \vee y$$

是Krom子句，其中只有最后一个子句不是Horn子句（第一个子句是Horn子句是因为$x \vee x = x$）。注意，一个Horn子句可包含任意数目的文字，而当限制于Krom子句时，本质上就是考虑2SAT问题。对于这两种情景，我们都将看到公式是否可满足是可在线性时间内加以判断的，即给定一个长为N的公式，只需简单执行$O(N)$个步骤即可。

首先考虑Horn子句。为什么它们如此容易处理？主要原因是一个如同$\bar{u} \vee \bar{v} \vee \bar{w} \vee \bar{x} \vee \bar{y} \vee z$的子句可改写为$\neg(u \wedge v \wedge w \wedge x \wedge y) \vee z$，它与

$$u \wedge v \wedge w \wedge x \wedge y \Rightarrow z$$

是一样的。换言之，如果u、v、w、x和y是真的，则z也必然是真的。由于这个原因，参数化的Horn子句被选为程序设计语言Prolog的基础机制。进一步，存在着简便的刻画那些布尔函数可以完全表示为Horn子句的描述。

定理H *布尔函数 $f(x_1, \cdots, x_n)$可表示为Horn子句的合取当且仅当对所有布尔值x_j和y_j*

$$f(x_1, \cdots, x_n) = f(y_1, \cdots, y_n) = 1 \text{ 蕴涵} f(x_1 \wedge y_1, \cdots, x_n \wedge y_n) = 1 \quad (33)$$

证明：[Alfred Horn，《J. Symbolic Logic》16(1951)，14—21，Lemma 7。] 若 $x_0 \vee \bar{x}_1 \vee \cdots \vee \bar{x}_k = 1$ 且 $y_0 \vee \bar{y}_1 \vee \cdots \vee \bar{y}_k = 1$，则

$$\begin{aligned} &(x_0 \wedge y_0) \vee \overline{x_1 \wedge y_1} \vee \cdots \vee \overline{x_k \wedge y_k} \\ &= (x_0 \vee \bar{x}_1 \vee \bar{y}_1 \vee \cdots \vee \bar{x}_k \vee \bar{y}_k) \wedge (y_0 \vee \bar{x}_1 \vee \bar{y}_1 \vee \cdots \vee \bar{x}_k \vee \bar{y}_k) \\ &\geqslant (x_0 \vee \bar{x}_1 \vee \cdots \vee \bar{x}_k) \wedge (y_0 \vee \bar{y}_1 \vee \cdots \vee \bar{y}_k) = 1 \end{aligned}$$

当上面没有横划线的x_0和y_0没有出现时，计算类似（但更简单）。因此每个Horn子句的合取都满足（33）。

反之，条件（33）则蕴涵f的每个基本子句是Horn子句（参见习题44）。

满足条件（33）的布尔函数，若还满足$f(1,\cdots,1)=1$，则称其是Horn函数（Horn function）或是确切的（definite）。易见，当且仅当每个子句恰好有一个没有上划线的文字时，Horn子句的合取是确切的，原因是如$\bar{x}\vee\bar{y}$这样的负子句，只有当所有的变量都为真时，整体才为假。确切Horn函数通常比Horn函数稍微容易处理些，因为前者显然是可满足的。因此，由定理H，存在唯一的使$f(x)=1$的最小向量$\boldsymbol{x}$，此即满足所有子句的所有向量的按位AND。一个确切Horn函数的核（core）是在此最小向量$\boldsymbol{x}$中为真的所有变量的集合。注意，当f为真时，核中的变量必是真的，因此，基本上我们能将其因子找出。

确切Horn函数会在多种场合出现，例如，在游戏分析中等（参见习题51和52）。另外一个美妙的例子来自于编译技术。考虑下面典型的（但非简化的）程序设计语言中代数表达式的语法。

⟨expression⟩→⟨term⟩|⟨expression⟩+⟨term⟩|⟨expression⟩−⟨term⟩
⟨term⟩→⟨factor⟩|−⟨factor⟩|⟨term⟩∗⟨factor⟩|⟨term⟩/⟨factor⟩
⟨factor⟩→⟨variable⟩|⟨constant⟩|(⟨expression⟩)
⟨variable⟩→⟨letter⟩|⟨variable⟩⟨letter⟩|⟨variable⟩⟨digit⟩　　(34)
⟨letter⟩→a | b | c
⟨constant⟩→⟨digit⟩|⟨constant⟩⟨digit⟩
⟨digit⟩→0 | 1

例如，字符串a/(-b0-10)+cc*cc符合⟨*expression*⟩的语法并且使用每条语法规则一次。

假定我们想知道在这样的表达式中哪两个字符是相邻出现的，使用确切Horn子句可给出问题的答案。原因是我们可以将问题表示为：令Xx、xX和xy表示“布尔命题”，其中X是集合{E, T, F, V, L, C, D}中的一个元素，分别表示⟨expression⟩，⟨term⟩，…，⟨digit⟩；x和y是集合{+, -, *, /, (,), a, b, c, 0, 1}中的符号。命题Xx意味着“X能以x结束”；类似地，xX意味着“X能以x开始”；而xy意味着“在一个表达式中，字符x后能直接跟随y”。（共有$7\times11+11\times7+11\times11=275$种可能性）。则我们有

xT⇒xE	⇒-T	xC⇒xF	Vx∧yL⇒xy	⇒Lc	
Tx⇒Ex	xF⇒-x	Cx⇒Fx	Vx∧yD⇒xy	xD⇒xC	
Ex⇒x+	Tx⇒x*	⇒(F	Dx⇒Vx	Dx⇒Cx	
xT⇒+x	xF⇒*x	xE⇒(x	⇒aL	Cx∧yD⇒xy	
Ex⇒x-	Tx⇒x/	Ex⇒x)	⇒La	⇒0D	(35)
xT⇒-x	xF⇒/x	⇒F)	⇒bL	⇒D0	
xF⇒xT	xV⇒xF	xL⇒xV	⇒Lb	⇒1D	
Fx⇒Tx	Vx⇒Fx	Lx⇒Vx	⇒cL	⇒D1	

其中，x和y取自11个终结符组成的集合{+, . . . , 1}。该示意性规范给出了总共$24\times11+3\times11\times11+13\times1=640$种确切Horn子句，如果对（35）中的⇒采用布尔

代数隐身记法⊖，则Horn子句可以形式化地写成

$$(\overline{+\mathrm{T}}\vee +\mathrm{E})\wedge(\overline{-\mathrm{T}}\vee -\mathrm{E})\wedge\cdots\wedge(\overline{\mathrm{V}+}\vee\overline{\mathrm{OL}}\vee +0)\wedge\cdots\wedge(\mathrm{D1})$$

为何要这样做呢？原因是所有这些子句的核是在此特别语法中为真的所有命题的集合。例如，可以验证-E为真，因此，符号（-在一个表达式中可以相互毗邻，而符号++和*-则不能（参见习题46）。

进一步，无需解决任何难题，我们就可以找出任何给定的确切Horn子句集合的核。我们从⇒左边为空的子句出发，即单独出现在⇒右边的命题开始；（35）中有13个此类子句。一旦确保了这些命题的真值，我们就可以发现更多的左边被确认为真的子句，因此其右边也属于核，过程以此方式持续执行。整个过程就像流水从山崖奔纵而下直到一定水平为止。事实上，若选择合适的数据结构，该过程很快，大约只需$O(N+n)$步，其中N是子句的总长度，n是命题变元的数目（这里假定所有的子句都已被扩充，而非如上使用参数x和y的缩略型，虽有成熟的可处理参数化子句的定理证明技术，但其超出了这里的讨论范围）。

算法C （确切Horn子句核的计算） 给定命题变量集合P和子句集合C，其中每个子句型为

$$u_1\wedge\cdots\wedge u_k\Rightarrow v \quad 其中\ k\geqslant 0\ 且\{u_1,\cdots,u_k,v\}\subseteq P \tag{36}$$

算法给出所有命题变量$Q\subseteq P$的集合，该命题变量集为真是所有子句为真的必要条件。

对于子句c、命题p、假设h（“假设”指的是出现在一个子句左边的命题），算法使用下面的数据结构。

CONCLUSION(c) 是子句c右边的命题；

COUNT(c) 是还没有得到确保的子句c的假设的数目；

TRUTH(p) 如果已知p为真，则为1，否则为0；

LAST(p) 是p所出现的最后一个假设；

CLAUSE(h) 是子句，其中h在其左边出现；

PREV(h) 是包含h命题的前一个假设。

算法还要维护一个已知为真但尚未被确保的所有命题S_0，S_1，…，S_{s-1}的栈。

C1. [初始化。] 对每个命题p令LAST(p)$\leftarrow\Lambda$，TRUTH(p)$\leftarrow 0$。令$s\leftarrow 0$，使栈为空。对每个型为（36）的子句c，令CONCLUSION(c)$\leftarrow v$，COUNT(c)$\leftarrow k$。如果$k=0$且TRUTH(v)=0，令TRUTH(v)$\leftarrow 1$，$S_s\leftarrow v$，$s\leftarrow s+1$；否则对于$1\leqslant j\leqslant k$，创建一个假设的记录h并令CLAUSE(h)$\leftarrow c$，PREV(h)$\leftarrow$LAST(u_j)，LAST(u_j)$\leftarrow h$。

C2. [准备确保p。] 如果$s=0$，算法终止，所期望的核包含了TRUTH值已被设为1的所有命题。否则，令$s\leftarrow s-1$，$p\leftarrow S_s$，$h\leftarrow$LAST(p)。

C3. [是否完成假设的处理？] 如果$h=\Lambda$，返回C2。

C4. [确认h有效。] 令$c\leftarrow$CLAUSE(h)，COUNT(c)$\leftarrow$COUNT(c)-1。如果COUNT(c)的新值仍然非0，转C6。

⊖ 也就是使用否定和析取连接符来表示⇒。——译者注

C5. [推导CONCLUSION(c)。] 令$p\leftarrow$CONCLUSION(c)。如果TRUTH(p)=0，令TRUTH(p)$\leftarrow 1$，$S_s\leftarrow p$，$s\leftarrow s+1$。

C6. [对h循环执行。] 令$h\leftarrow$PREV(h)，返回C3。▮

值得注意的是，算法使用了协作流畅的数据结构，由此避免了为了取得计算进展而需要的对位置进行的搜索。算法C在许多方面和算法2.2.3.T（拓扑排序）相类似，后者出现在第2章，是很早就讨论过的多链数据结构的首个例子；事实上，如果每个命题恰好出现在一个子句的右边，那么算法2.2.3.T就可以看成算法C的特例（参见习题47）。

习题48说明了对算法C进行少许修改就可以用于解决一般的Horn子句的可满足性问题。进一步的讨论见W.F. Dowling和J.H.Gallier的文章《J. Logic Programming》1 (1984)，267—284。

现在转向Krom 函数和2SAT问题。对此，也有线性时间的算法。先看一个简单而实际的应用，也许它能帮助我们更好地去欣赏算法。假定在3天的喜庆日中，有7位喜剧演员，每个都同意参加5个宾馆中2个的一晚表演，但由于其他合约在身，每位演员在这3天中只有2天有空。

Tomlin 应第1天和第2天在Aladdin和Caesars演出；
Unwin应第1天和第2天在Bellagio和Excalibur演出；
Vegas 应第2天和第3天在Desert 和Excalibur演出；
Williams 应第1天和第3天在Aladdin和 Desert演出； (37)
Xie应第1天和第3天在 Caesars 和Excalibur演出；
Yankovic应第2天和第3天在 Bellagio和 Desert演出；
Zany应第1天和第2天在 Bellagio 和Caesars演出。

能给出没有冲突的演出安排吗？

为了解决这个问题，我们引入7个布尔变量$\{t, u, v, w, x, y, z\}$，其中，比如说，t的意思是Tomlin第1天在Aladdin且第2天在Caesars，而$\bar{t}$的意思是演出宾馆的次序相反。接着我们就可以安排这些喜剧演员的演出日程以保证没有2个喜剧演员同一天被安排在同一宾馆演出。

$$\begin{array}{llll}\neg(t\wedge m)[\mathrm{A1}] & \neg(y\wedge \bar{z})[\mathrm{B2}] & \neg(t\wedge z)[\mathrm{C2}] & \neg(w\wedge y)[\mathrm{D3}]\\ \neg(u\wedge z)[\mathrm{B1}] & \neg(\bar{t}\wedge x)[\mathrm{C1}] & \neg(v\wedge \bar{y})[\mathrm{D2}] & \neg(\bar{u}\wedge \bar{x})[\mathrm{E1}]\\ \neg(\bar{u}\wedge y)[\mathrm{B2}] & \neg(\bar{t}\wedge \bar{z})[\mathrm{C1}] & \neg(\bar{v}\wedge w)[\mathrm{D3}] & \neg(u\wedge \bar{v})[\mathrm{E2}]\\ \neg(\bar{u}\wedge \bar{z})[\mathrm{B2}] & \neg(x\wedge \bar{z})[\mathrm{C1}] & \neg(\bar{v}\wedge y)[\mathrm{D3}] & \neg(v\wedge x)[\mathrm{E3}]\end{array} \tag{38}$$

当然，这里的每个约束都是Krom子句，我们必须满足下面的公式。

$$\begin{aligned}&(\bar{t}\vee\bar{w})\wedge(\bar{u}\vee\bar{z})\wedge(u\vee\bar{y})\wedge(u\vee z)\wedge(\bar{y}\vee z)\wedge(t\vee\bar{x})\wedge(t\vee z)\wedge(\bar{x}\vee z)\\&\wedge(\bar{t}\vee\bar{z})\wedge(\bar{v}\vee y)\wedge(v\vee\bar{w})\wedge(v\vee\bar{y})\wedge(\bar{w}\vee\bar{y})\wedge(u\vee x)\wedge(\bar{u}\vee v)\wedge(\bar{v}\vee\bar{x})\end{aligned} \tag{39}$$

进一步，Krom子句（和Horn子句类似）可以被写成蕴涵式。

$$\begin{array}{llllllll}t\Rightarrow\bar{w}, & u\Rightarrow\bar{z}, & \bar{u}\Rightarrow\bar{y}, & \bar{u}\Rightarrow z, & y\Rightarrow z, & \bar{t}\Rightarrow\bar{x}, & \bar{t}\Rightarrow z, & x\Rightarrow z\\ t\Rightarrow\bar{z}, & v\Rightarrow y, & \bar{v}\Rightarrow\bar{w}, & \bar{v}\Rightarrow\bar{y}, & w\Rightarrow\bar{y}, & \bar{u}\Rightarrow x, & u\Rightarrow v, & v\Rightarrow\bar{x}\end{array} \tag{40}$$

每个这样的蕴涵式都有一个与之等价的表达式，即其“逆反”式。

$$\begin{matrix} w\Rightarrow\bar{t}, & z\Rightarrow\bar{u}, & y\Rightarrow u, & \bar{z}\Rightarrow u, & \bar{z}\Rightarrow\bar{y}, & x\Rightarrow t, & \bar{z}\Rightarrow t, & \bar{z}\Rightarrow\bar{x} \\ z\Rightarrow\bar{t}, & \bar{y}\Rightarrow\bar{v}, & w\Rightarrow v, & y\Rightarrow v, & y\Rightarrow\bar{w}, & \bar{x}\Rightarrow u, & \bar{v}\Rightarrow\bar{u}, & x\Rightarrow\bar{v} \end{matrix} \tag{41}$$

哦，这里有一个恶劣的循环

$$u \underset{[B1]}{\Rightarrow} \bar{z} \underset{[B2]}{\Rightarrow} \bar{y} \underset{[D2]}{\Rightarrow} \bar{v} \underset{[E2]}{\Rightarrow} \bar{u} \underset{[B2]}{\Rightarrow} z \underset{[C2]}{\Rightarrow} \bar{t} \underset{[C1]}{\Rightarrow} \bar{x} \underset{[E1]}{\Rightarrow} u\ . \tag{42}$$

该循环说明u和$\bar{u}$必须有相同的值；因此（37）中的所有条件无法同时满足。为了得到可行的演出安排，节日组织者不得不和6个喜剧演员$\{t,u,v,w,y,z\}$中的至少一位协商演出合同（参见习题53）。

例如，组织者可以尝试临时将v留在图的外面。此时（38）中的6个约束将去掉5个，而（40）和（41）中只留下了22个蕴涵式，这样就得到了如图6所示的有向图。该有向图也包含循环，如$z\Rightarrow\bar{u}\Rightarrow x\Rightarrow z$和$t\Rightarrow\bar{z}\Rightarrow t$；但没有一个循环同时包含一个变量和它的补。实际上，从图6我们可以看到，值$tuwxyz$=110000满足（39）中不涉及v或$\bar{v}$的每个子句。这些值给出了满足（37）所描述的7个演员初始契约中6个的演出安排，即第一天的安排是(Tomlin, Unwin, Zany, Williams, Xie）在（Aladdin, Bellagio, Caesars, Desert, Excalibur)。

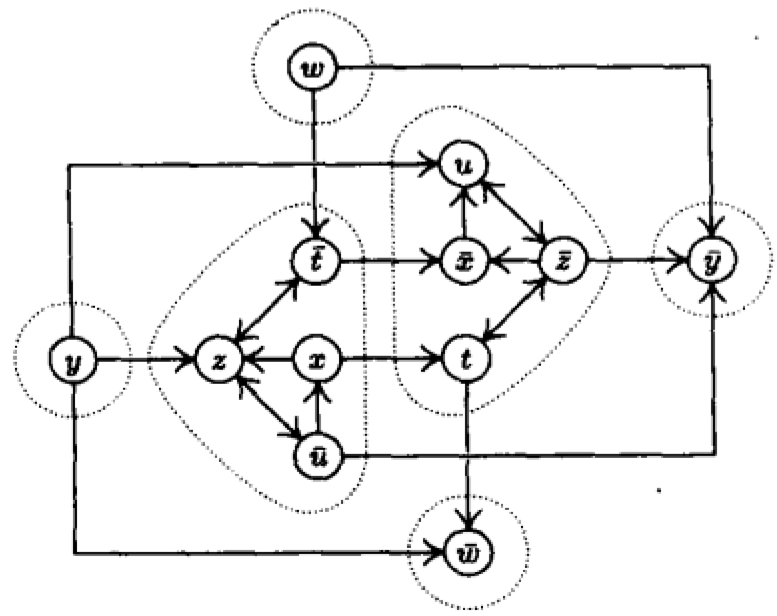


图6 对应（40）和（41）但不包含v或$\bar{v}$的有向图

注：对每个强连通分量中的文字进行合适的赋值就可以解决二元调度问题，该问题是2SAT的一个特例。

一般地，给定有m个Krom子句、含n个命题变元的一个2SAT问题，我们都可以按同样的方式构造一个有向图。该有向图有$2n$个顶点$\{x_1, \bar{x}_1, \cdots, x_n, \bar{x}_n\}$，每个顶点对应一个可能的文字；有$2m$条形为$\bar{u}\to v$和$\bar{v}\to u$的弧，对每个子句$u\vee v$有两条弧。在该有向图中，当且仅当存在从$u$到$v$和从$v$到$u$的有向路径时，文字$u$和$v$属于同一强连通分量。例如，图6中的有向图有6个由点轮廓标出的强连通分量。对于任何相应的2SAT问题解，一个强连通分量中的所有文字必须有相同的布尔值。

定理K 每个子句含2个文字的合取范式是可满足的，当且仅当其相应的有向图中没有同时包含一个变量和其补的强连通分量。

证明：[Melven Krom，《Zeitschrift für mathematische Logik und Grundlagen der Mathematik》13（1967），15－20，Corollary 2.2。]如果存在x到$\bar{x}$和$\bar{x}$到x的路径，公式肯定是不可满足的。

反之，假定不存在这样的路径。任何一个有向图至少有一个强连通分量S，称为“源”，它没有来自其他强连通分量顶点的弧。进一步，我们的有向图如图6所示，有着吸引人的反对称性：$u\to v$当且仅当$\bar{v}\to\bar{u}$。因此S中文字的补形成了另一个强连通分量$\bar{S}\neq S$，可称为“汇”，它没有指向其他强连通分量的弧。因此，可对S中

的文字赋予值1，而对$\overline{S}$中的文字赋予值0，然后将它们从有向图中移去，并以相同的方式继续处理直到所有文字都被赋值为止。当$u \to v$在有向图中时，上述过程得到的值满足$u \leqslant v$；因此当$\bar{u} \vee v$是公式的一个子句时，满足$\bar{u} \vee v$。

基于R. E. Tarjan给出的在线性时间内找出强连通分量的算法，由定理K直接可得2SAT问题的一个有效解决方案[参见《SICOMP》1 (1972)，146－160；D.E. Knuth，《The Stanford GraphBase》(1993)，512－519]。7.4.1节将详细研究Tarjan的算法，习题54表明，当算法检测到一个新的强连通分量时，定理K的条件可被快速核查。进一步，算法先检测到“汇”，因此作为Tarjan算法的一个简单副作用，我们可以对每个比其补先出现在一个强连通分量的文字指派值1来使可满足性成立。

中数 前面主要讨论二元布尔运算，如$x \vee y$或$x \oplus y$。但也存在着一种重要的、称为x、y和z中数（median）的三元（ternary）运算$\langle xyz \rangle$：

$$\langle xyz \rangle = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z) = (x \vee y) \wedge (y \vee z) \wedge (x \vee z) \tag{43}$$

事实上，$\langle xyz \rangle$可能是整个布尔领域最为重要的三元运算，它的令人着迷的性质一直在被发现或重复被发现。

首先，容易看出公式$\langle xyz \rangle$描述了任何布尔量x、y和z的众值（majority）：$\langle 000 \rangle = \langle 001 \rangle = 0$和$\langle 011 \rangle = \langle 111 \rangle = 1$。若$x$、$y$和$z$是任意的实数，而$\wedge$和$\vee$分别表示min和max，则

$$\langle xyz \rangle = y \quad 当x \leqslant y \leqslant z时 \tag{44}$$

因此，称$\langle xyz \rangle$为“中数”而非“众数”。

接着，基本二元操作$\wedge$和$\vee$都是中数函数的特例为

$$x \wedge y = \langle x0y \rangle; \quad x \vee y = \langle x1y \rangle \tag{45}$$

因此，任意的单调布尔函数都可以完全借助三元中数运算以及常量0和1表示。事实上，在一个仅有中数的世界里，可以令$\wedge$表示假而$\vee$表示真，则$x \wedge y = \langle x \wedge y \rangle$和$x \vee y = \langle x \vee y \rangle$将是完美且自然的表达式，愿意的话，甚至可以使用波兰表示法，即$\langle \wedge xy \rangle$和$\langle \vee xy \rangle$。若就常量$\wedge = -\infty$和$\vee = +\infty$讨论中数，同样的思想也适用于使用$\wedge$和$\vee$表示min和max的实数扩展。

若布尔函数$f(x_1, x_2, \cdots, x_n)$满足

$$\overline{f(x_1, x_2, \cdots, x_n)} = f(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n) \tag{46}$$

则称其是自对偶（self-dual）的。

我们已经知道一个布尔函数是单调的当且仅当它可以使用$\wedge$和$\vee$表示，由De Morgan定律（11）和（12）可知，一个单调公式是自对偶的当且仅当$\wedge$和$\vee$互换而公式的值不会改变。因此由（43）定义的中数操作既是单调的又是自对偶的。事实上，这是此类中的最简单的非平凡函数，因为除了投影函数 L 和 R 外，表1中没有一个二元操作既是单调的又是自对偶的。

进一步，不使用常量而完全由中数操作构造的任何表达式既是单调的又是自对偶的。例如，函数$\langle w \langle xyz \rangle \langle w \langle uvw \rangle x \rangle \rangle$是自对偶的，这是因为

$$\overline{\langle w\langle xyz\rangle\langle w\langle uvw\rangle x\rangle\rangle}=\langle \overline{w}\overline{\langle xyz\rangle}\overline{\langle w\langle uvw\rangle x\rangle}\rangle$$
$$=\langle \overline{w}\langle \overline{x}\overline{y}\overline{z}\rangle\langle \overline{w}\overline{\langle uvw\rangle}\overline{x}\rangle\rangle=\langle \overline{w}\langle \overline{x}\overline{y}\overline{z}\rangle\langle \overline{w}\langle \overline{u}\overline{v}\overline{w}\rangle\overline{x}\rangle\rangle$$

Emil Post在他的博士论文（哥伦比亚大学，1920）中证明了其反向也为真。

定理P *每个单调、自对偶布尔函数$f(x_1, x_2, \cdots, x_n)$可完全由中数操作$\langle xyz\rangle$表示。*

证明：[《Annals of Mathematics Studies》5 (1941), 74—75。] 首先考察公式

$$\langle x_1 y\langle x_2 y\cdots y\langle x_{s-1} y x_s\rangle\cdots\rangle\rangle$$
$$=((x_1\vee x_2\vee\cdots\vee x_{s-1}\vee x_s)\wedge y)\vee(x_1\wedge x_2\wedge\cdots\wedge x_{s-1}\wedge x_s) \qquad (47)$$

该公式是重复应用中数操作的结果，对s进行归纳就可证明之。

假定$f(x_1, x_2, \cdots, x_n)$是单调、自对偶的，并具下述析取基本型

$$f(x_1,\cdots,x_n)=t_1\vee\cdots\vee t_m,\quad t_j=x_{j1}\wedge\cdots\wedge x_{js_j}$$

其中没有一个基本蕴涵项t_j被包含在另一蕴涵项中（推论Q）。易见，任何两个基本蕴涵项都至少有一共同变量。这是因为，假定我们有$t_1=x\wedge y$，$t_2=u\wedge v\wedge w$，则当$x=y=1$且$u=v=w=0$，或$x=y=0$且$u=v=w=1$时，f的值是1，这和题设的自对偶性相矛盾。因此，若某t_j仅包含一个变量x，它必是唯一的此类基本蕴涵项，在这种情况下，f是平凡函数$f(x_1, x_2, \cdots, x_n)=x=\langle xxx\rangle$。

通过中数操作的复合应用，我们可以定义函数$g_0, g_1, \cdots, g_m$如下。

$$g_0(x_1,\cdots,x_n)=x_1$$
$$g_j(x_1,\cdots,x_n)=h(x_{j1},\cdots,x_{js_j};\ g_{j-1}(x_1,\cdots,x_n)),\ 对于1\leqslant j\leqslant m \qquad (48)$$

这里$h(x_1, \cdots, x_s; y)$是等式（47）左方的函数。因为当$k<j$时$(x_{j1}\vee\cdots\vee x_{js_j})\wedge t_k=t_k$，对$j$进行归纳，就可以由（47）和（48）证明结论：当$t_1\vee\cdots\vee t_j=1$时，$g_j(x_1, x_2, \cdots, x_n)=1$。

最后，$f(x_1, x_2, \cdots, x_n)$必等于$g_m(x_1, x_2, \cdots, x_n)$，原因是它们都是单调和自对偶的，且我们已经证明对所有0和1的组合，$f(x_1, x_2, \cdots, x_n)\leqslant g_m(x_1, x_2, \cdots, x_n)$。因为一个自对偶函数恰好在$2^n$可能的情况中有一半是等于1，因此由该不等式就可以证明等式成立。▮

定理P的结论之一是可以使用3个参数的中数操作来表示5个元素的中数操作，原因是任意奇数个布尔变量的中数操作一定是单调和自对偶的布尔操作。若记这样的中数操作为$\langle x_1\cdots x_{2k-1}\rangle$，则$\langle vwxyz\rangle$的析取基本型为

$$(v\wedge w\wedge x)\vee(v\wedge w\wedge y)\vee(v\wedge w\wedge z)\vee(v\wedge x\wedge y)\vee(v\wedge x\wedge z)$$
$$\vee(v\wedge y\wedge z)\vee(w\wedge x\wedge y)\vee(w\wedge x\wedge z)\vee(w\wedge y\wedge z)\vee(x\wedge y\wedge z)$$

因此在定理P证明的构造中，$\langle vwxyz\rangle$被表示成一个包含2046个3元中数操作的庞大公式$g_{10}(v, w, x, y, z)$。当然该表达式可能并非最短，实际上，我们有

$$\langle vwxyz\rangle=\langle v\langle xyz\rangle\langle wx\langle wyz\rangle\rangle\rangle \qquad (49)$$

[参见 H. S. Miiller和R. O. Winder，《IRE Transactions》 EC-11 (1962),89—90。]

***中数代数和中数图** 如前所述，当x、y和z属于任意的有序集，如实数集，而$\wedge$和$\vee$被看作min和max操作时，三元操作$\langle xyz\rangle$是有用的。事实上，在更一般的情况下，操作$\langle xyz\rangle$也扮演着有意义的角色。中数代数（median algebra）是任意集合M及其上所定义的三元操作$\langle xyz\rangle$，该操作将M中的元素映射为M中的元素并且满足下面3条公理

$$\langle xxy\rangle=x\text{(众数律)} \tag{50}$$

$$\langle xyz\rangle=\langle xzy\rangle=\langle yxz\rangle=\langle yzx\rangle=\langle zxy\rangle=\langle zyx\rangle\ \text{(交换律)} \tag{51}$$

$$\langle xw\langle ywz\rangle\rangle=\langle\langle xwy\rangle wz\rangle\text{(结合律)} \tag{52}$$

例如，在布尔情景下，因为$\wedge$和$\vee$操作是可结合的，对于$w=0$和$w=1$结合律（52）成立。习题75和76证明了上述3条公理也蕴涵着中数操作是满足分配律（distributive law）的，该分配律有一个短的形式

$$\langle\langle xyz\rangle uv\rangle=\langle x\langle yuv\rangle\langle zuv\rangle\rangle \tag{53}$$

和一个更为对称的长的形式

$$\langle\langle xyz\rangle uv\rangle=\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle \tag{54}$$

目前还未知晓该事实是否存在简单的证明，但至少对于$y=u$和$z=v$这一简单情景，我们可以验证式（53）和（54）成立。因为等式两边都等于$\langle xy\langle zyz\rangle\rangle$，由此

$$\langle\langle xyz\rangle yz\rangle=\langle xyz\rangle \tag{55}$$

事实上结合律（52）仅仅是（53）在$y=u$时的特殊情况。基于（55）和（52），我们还可以验证

$$x=u:\langle\langle xyz\rangle uv\rangle=\langle vu\langle yuz\rangle\rangle=\langle\langle vuy\rangle uz\rangle=\langle\langle yuv\rangle uz\rangle=\langle\langle\langle yuv\rangle uv\rangle uz\rangle=$$
$$\langle\langle yuv\rangle u\langle vuz\rangle\rangle=\langle u\langle yuv\rangle\langle zuv\rangle\rangle$$

中数代数M的理想（ideal）是子集$C\subseteq M$，有性质

$$\text{当}x\in C,\ y\in C\text{且}z\in M\text{时}\ \langle xyz\rangle\rightarrow\in C \tag{56}$$

如果u和v是M的任意元素，则可定义区间$[u..v]$如下：

$$[u..v]=\{\langle xuv\rangle\mid x\in M\} \tag{57}$$

当且仅当$x\in[u..v]$，我们称“x在u和v之间”。由定义知，u和v自身总是属于区间$[u..v]$的。

引理M 每个区间$[u..v]$是一个理想，且$x\in[u..v]\Leftrightarrow x=\langle uxv\rangle$。

证明：令$\langle xuv\rangle$和$\langle yuv\rangle$是$[u..v]$中的任意元素。由（51）和（53），对所有$z\in M$，有

$$\langle\langle xuv\rangle\langle yuv\rangle z\rangle=\langle\langle xyz\rangle uv\rangle\in[u..v]$$

因此，$[u..v]$是一个理想。进一步，由（51）和（55），每个元素$\langle xuv\rangle\in[u..v]$都满足$\langle xuv\rangle=\langle u\langle xuv\rangle v\rangle$。■

因为有中数定律，这里定义的区间有着很好的性质。

$$v \in [u..u] \Rightarrow u=v \tag{58}$$

$$x \in [u..v] \text{和} y \in [u..x] \Rightarrow y \in [u..v] \tag{59}$$

$$x \in [u..v] \text{和} y \in [u..x] \text{和} y \in [v..z] \Rightarrow y \in [x..z] \tag{60}$$

等价地，$[u..u]=\{u\}$；如果$x\in[u..v]$，则$[u..x]\subseteq[u..v]$；且$x\in[u..v]$也蕴涵着对所有z有$[u..z]\cap[v..z]\subseteq[x..z]$（参见习题72）。

现在让我们在顶点集合M上定义一个图，其边为

$$u — v \Leftrightarrow u\neq v \text{ 且对所有} x\in M \text{ 有 } \langle xuv\rangle\in\{u,v\} \tag{61}$$

换句话说，当且仅当区间$[u..v]$仅仅包含两个点u和v，那么u和v是邻接顶点。

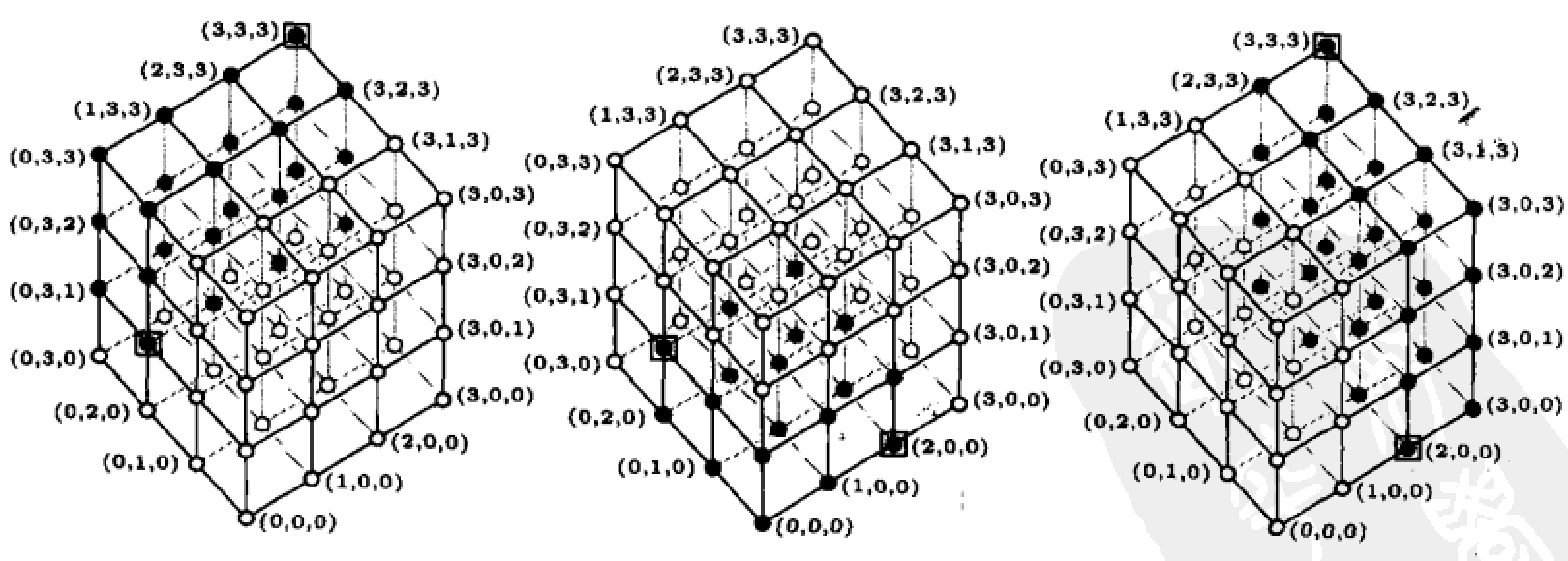
定理G 如果M是任意的有穷中数代数，则由（61）定义的图是连通的。进一步，当且仅当顶点x位于从u到v的最短路径上，x属于区间$[u..v]$。

证明：如果M不是连通的，选择两个顶点u和v，其间不存在路径且区间$[u..v]$有尽可能少的元素。令$x\in[u..v]$且不同于u和v，则$\langle xuv\rangle=x\neq v$，因此$v\notin[u..x]$；类似地有$u\notin[x..v]$。但由（59），$[u..x]$和$[x..v]$包含于$[u..v]$，因此它们是更小的区间，且必有一个从$u$到$x$和从$x$到$v$的路径。矛盾。

定理的另一半的证明参见习题73。▮

由（55）有$\langle xyz\rangle=\langle\langle xyz\rangle xy\rangle=\langle\langle xyz\rangle xz\rangle=\langle\langle xyz\rangle yz\rangle$，因此这里定义的区间蕴涵着$\langle xyz\rangle\in[x..y]\cap[x..z]\cap[y..z]$。反之，如果$w\in[x..y]\cap[x..z]\cap[y..z]$，习题74将证明$w=\langle xyz\rangle$。换句话说，当$x$、$y$和$z$是$M$中的点时，交集$[x..y]\cap[x..z]\cap[y..z]$总是恰好包含着一个点。

图7基于一个4×4×4的立方体说明了上述原理，立方体中的每个点x的坐标为(x_1, x_2, x_3)，其中$0\leqslant x_1, x_2, x_3<4$。由于$\langle xyz\rangle=(\langle x_1y_1z_1\rangle, \langle x_2y_2z_2\rangle, \langle x_3y_3z_3\rangle)$，因此该立方体的顶点形成了一个中数代数。进一步，图7中的边就是（61）所定义的那些边，所连接的2个顶点的坐标除了一个坐标值相差±1外，其他相同。如图所示，3个典型的区间是$[x..y]$、$[x..z]$和$[y..z]$，3个区间仅有的公共点是顶点$\langle xyz\rangle=(2, 2, 1)$。



a) 区间$[x..y]$　　b) 区间$[x..z]$　　c) 区间$[y..z]$

图7　在一个4×4×4的立方体中顶点$x=(0, 2, 1)$、$y=(3, 3, 3)$和$z=(2, 0, 0)$之间的区间

到目前为止，我们都是从中数代数出发，并使用它定义了一个有着某种性质的图。但我们也可以从具备相应性质的图出发来定义中数代数。如果u和v是任意

一个图的顶点，我们可定义区间$[u..v]$是u和v之间最短路径上的所有顶点的集合。如果在连接任意给定的顶点x、y和z的3个区间的交集$[x..y]\cap[x..z]\cap[y..z]$中恰好只有一个顶点，则称图是中数图（median graph），并将相应的顶点记为$\langle xyz\rangle$。习题75证明这样产生的3元操作满足中数公理。

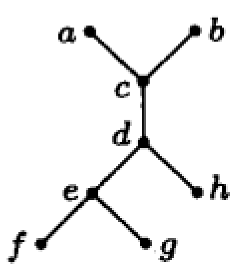
按照上述定义，许多重要的图都是中数图。例如，容易看出，任意自由树都是中数图，而如$n_1\times n_2\times\cdots\times n_m$这样的超矩形则是另一个例子。另外，任意中数图的笛卡儿积也满足中数图所需要具备的条件。

***中数标记** 如果u和v是一个中数代数中的任意元素，由长形式的分配律（54）可知，映射$f(x)$：$x\mapsto\langle xuv\rangle$是一个同态映像，即其满足

$$f(\langle xyz\rangle)=\langle f(x)f(y)f(z)\rangle \tag{62}$$

由（57），函数$\langle xuv\rangle$将任何给定的点x投影到区间$[u..v]$。当u—v是相应图的一条边时特别有意思，因为此时$f(x)$是2值的，因此本质上是一个布尔映射。

例如，考虑下面所示的有8个顶点、7条边的自由树。通过判断x是与u或v相近，可将每个顶点x映射到每条边对应的区间$[u..v]$之中。



		ac	bc	cd	de	ef	eg	dh		
a	$\mapsto$	a	c	c	d	e	e	d	0000000	
b	$\mapsto$	c	b	c	d	e	e	d	1100000	
c	$\mapsto$	c	c	c	d	e	e	d	1000000	
d	$\mapsto$	c	c	d	d	e	e	d	1010000	(63)
e	$\mapsto$	c	c	d	e	e	e	d	1011000	
f	$\mapsto$	c	c	d	e	f	e	d	1011100	
g	$\mapsto$	c	c	d	e	e	g	d	1011010	
h	$\mapsto$	c	c	d	d	e	e	h	1010001	

在图的右边，我们将投影约简为0和1，其中$a\mapsto 0000000$是随意确定的，这样所导致的位串称为顶点的标记，可写成如$l(b)=1100000$这样的形式。因为每个投影都是一个同态映像，通过对标记中的每个成分取布尔中数就能计算出任意3点的中数。例如，要计算$\langle bgh\rangle$，只需找出$l(b)=1100000$、$l(g)=1011010$和$l(h)=1010001$的按位中数即可，该值为$1010000=l(d)$。

当投影到一个中数图的所有边时，我们可能会发现二进制标记中有2个列是相等的。对于自由树则不会发生这种情况，但考虑一下，如果往图（63）树中添加边g—h会发生什么：所产生的图仍是一个中数图，但列eg和dh将变成相等（除了$e\leftrightarrow d$和$g\leftrightarrow h$）。进一步，gh产生的新列将和de对应的列相等。在这种情况下，冗余的成分应从标记中略去，因此扩充图的顶点标记将是6位而不是7位，如$l(g)=101101$、$l(h)=101001$。

任意中数代数的元素都可以依此方式表示为标记。因此任意布尔情景下的等式在中数代数中亦成立。借助此“0-1原理”，测试任意2个由三元操作$\langle xyz\rangle$构造的表达式在公理（50）、（51）和（52）下是否相等成为可能，尽管依照这里给出的方法，在测试含n个变元的表达式时，不得不检查$2^{n-1}-1$种情况。

例如，结合律$\langle xw\langle ywz\rangle\rangle=\langle\langle xwy\rangle wz\rangle$提示了等式两边都应有一个对称的不涉及嵌

套括号的解释，而实际上，也存在着这样的公式：

$$\langle xw\langle ywz\rangle\rangle=\langle\langle xwy\rangle wz\rangle=\langle xwywz\rangle \qquad (64)$$

这里$\langle xwywz\rangle$表示5个元素的多重集$\{x, w, y, w, z\}=\{w, w, x, y, z\}$的中数。我们可以使用0-1原理来证明该公式，也请注意，中数等同于布尔情景下的众数。以类似的方式，我们可以证明（49），也可以将Post在（47）中使用的函数简化为

$$\langle x_1y\langle x_2y\cdots y\langle x_{s-1}yx_s\rangle\cdots\rangle\rangle=\langle x_1yx_2y\cdots yx_{s-1}yx_s\rangle \qquad (65)$$

它是$2s-1$个量的中数，其中几乎有一半的数等于y。

若$u\in C$且$v\in C$时$[u\,.\,.\,v]\subseteq C$，则称图中的顶点集C为凸的（convex）。换句话说，就是当一条最短路径的端点属于C时，该路径上的所有顶点均出现在C中。（因此凸集就等价于前文所介绍的“理想”，而现在我们使用的语言就从代数术语变成几何术语了。）$\{v_1, \cdots, v_m\}$ 的凸包（convex hull）定义为包含$v_1, \cdots, v_m$中每个顶点的最小凸集。上面理论结果表明每个区间$[u\,.\,.\,v]$都是凸的，因此$[u\,.\,.\,v]$是两点集$\{u, v\}$的凸包。但事实上还有很多其他的性质。

定理C 在一个中数图中，$\{v_1, v_2, \cdots, v_m\}$的凸包是下面所有点的集合。

$$C=\{\langle v_1xv_2x\cdots xv_m\rangle \mid x\in M\} \qquad (66)$$

进一步，当且仅当$x=\langle v_1xv_2x\cdots xv_m\rangle$，$x$在$C$中。

证明：显然，对所有$1\leqslant j\leqslant m$，$v_j\in C$。因为点$x'=\langle v_2x\cdots xv_m\rangle$在凸包中（对$m$进行归纳可证）且$\langle v_1x\cdots xv_m\rangle\in[v_1\,.\,.\,x']$，因此$C$中的每个点必属于凸包。由0-1原理可证：

$$\langle x\langle v_1yv_2y\cdots yv_m\rangle\langle v_1zv_2z\cdots zv_m\rangle\rangle=\langle v_1\langle xyz\rangle v_2\langle xyz\rangle\cdots\langle xyz\rangle v_m\rangle \qquad (67)$$

因此C是凸的。在公式中令$x=y$就可证得$\langle v_1xv_2x\cdots xv_m\rangle$是$C$中和$x$最近的点，且对所有$z\in C$有$\langle v_1xv_2x\cdots xv_m\rangle\in[x\,.\,.\,z]$。▌

推论C 对所有$1\leqslant j\leqslant m$，令v_j的标记为$v_{j1}\cdots v_{jt}$，则$\{v_1, v_2, \cdots, v_m\}$的凸包是所有$x\in M$的集合，当$v_{1j}=v_{2j}=\cdots=v_{mj}=c_j$时标记$x_1\cdots x_t$满足$x_j=c_j$。▌

例如，（63）中$\{c, g, h\}$的凸包包含了标记可以和模式10**0**相匹配的所有元素，即$\{c, d, e, g, h\}$。

当中数图包含一个4循环$u—x—v—y—u$时，因为投影到$[u\,.\,.\,x]$和投影到$[v\,.\,.\,y]$都产生同一标记坐标，因此就此而言，边$u—x$和$v—y$是等价的。原因是，对任何满足$\langle zux\rangle=u$的z，我们有

$$\begin{aligned} y=\langle uvy\rangle &=\langle\langle zux\rangle vy\rangle \\ &=\langle\langle zvy\rangle\langle uvy\rangle\langle xvy\rangle\rangle \\ &=\langle\langle zvy\rangle yv\rangle \end{aligned}$$

因此$\langle zvy\rangle=y$；类似地$\langle zux\rangle=x$蕴涵$\langle zvy\rangle=v$。基于同样的理由，边$x—v$和$y—u$也是等价的。除此之外，习题77还表明，两条边产生的投影是等价的，当且仅当以此方式由4循环所得的等价链可以证明它们是等价的。因此，每个顶点的标记位数等于由4循环所诱导的等价类的数目，且一旦指定了一个顶点的标记为00…0，顶点的精简标记也是唯一确定的。

P. K. Jha和G. Slutzki [《Ars Combin》. 34(1992), 75－92]给出了一个寻找中数图顶点标记的漂亮方法，该方法由J. Hagauer、W. Imrich和S. Klavžar加以改进[《Theor. Comp. Sci.》215(1999)，123－136]。

算法H（中数标记，median label） 给定一个中数图G和一个源顶点a，本算法确定由G的4循环定义的等价类并计算每个顶点的标记$l(v)=v_1\cdots v_t$，其中t是等价类的数目且$l(a)=0\cdots0$。

H1.[初始化。] 对图进行预处理，以与a的距离为序访问图中的所有顶点。对每条边u—v，若a和u的距离比a和v的距离近，则称u是v的早期邻居（early neighbour），否则称u是v的晚期邻居（later neighbour）邻居。换句话说，v的早期邻居在处理v之前就已经被访问过了，而晚期邻居则仍等待着被访问。重排所有的邻接表以使早期邻居先被列出。初始时将每条边放置于属于自己的等价类之中，而后一旦得知两个类等价，就使用类似于算法2.3.3E的“union-find算法”来归并它们。

H2.[调用子例程。] 令$j\leftarrow0$并使用参数a调用子例程I。（子例程I在后面列出。其中全局变量j被用来创建边r_j—s_j的主导表（master list），这里$1\leqslant j<n$，n是顶点的数目，对每个$v\neq a$的顶点v，表中有元素$s_j=v$。）

H3.[指派标记。] 从1到t对等价类进行标识。令$l(a)$是t位串$0\cdots0$。对于$j=1$，2，…，$n-1$（以此为序），令$l(s_j)$为$l(r_j)$，其第k位已从1改为0，这里k是边r_j—s_j的等价类。

子例程I（处理r的后代）。本递归子例程的参数为r，使用全局变量j完成算法H的主要工作，处理当前从r出发可达的图中所有顶点。在处理过程中，除了r本身之外的所有此类顶点都被记录在主导表（master list）中；而在对应的当前图中，此类顶点间的边均被移去。子例程中的每个顶点有4个域，LINK、MARK、RANK和MATE初始皆为空。

I1.[对s循环。] 选择一个满足r—s的顶点s。如果没有这样的顶点，从子例程返回。

I2.[记录边。] 令$j\leftarrow j+1$，$r_j\leftarrow r$且$s_j\leftarrow s$。

I3.[开始广度优先搜索。]（现要找出并删除当前图中所有与r—s等价的边）。令MARK(s)←s，RANK(s)←1，LINK(s)←Λ且$v\leftarrow q\leftarrow s$。

I4.[找出v的配对。] 寻找MARK(u)≠s的v的早期邻居u。（恰好存在一个这样的u。回忆一下，在步骤H1中，早期邻居已被放置好了。）令MATE(v)←u。

I5.[删除u—v。] 通过归并等价类使边u—v和r—s等价。将u和v从各自的邻接表中移去。

I6.[对v的邻居分类。] 对v的每个早期邻居u执行I7，对v的每个晚期邻居u执行I8。接着转向I9。

I7.[记下可能的等价。] 如果MARK(u)=s且RANK(u)=1，使边u—v和边MATE(u)—MATE(v)等价。返回到I6。

I8.[设置u的RANK值。] 如果MARK(u)=s且RANK(u)=1，返回到I6。否则令

MARK(u)←s且RANK(u)←2。令w是u的第一个邻居（它总是早期的）。如果$w=v$，重置w为u的第二个早期邻居；但若u只有一个早期邻居则返回到I6。如果MARK(w)≠s或RANK(w)≠2，令RANK(u)←1，LINK(u)←Λ，LINK(q)←u且q←u。返回到I6。

I9. [继续广度优先搜索。] 令v←LINK(v)。如果v≠Λ，则返回到I4。

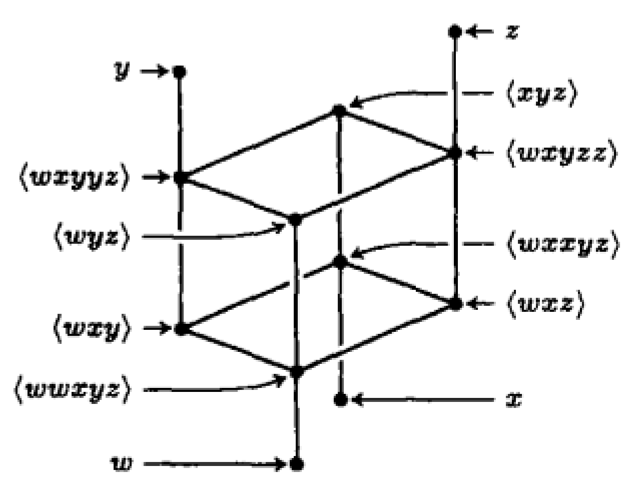
I10. [处理子图s。] 使用参数s递归调用子例程I。然后返回到I1。▮

本算法和子例程使用相对高级的数据结构进行描述，进一步的细节留给读者。例如，邻接表应该是双链的，这样步骤I5中删除边的操作就可方便进行。在该步可使用任何归并等价类的方便之法。

习题77给出算法正确性的理论解释，习题78证明了在步骤I4中每个顶点最多被遇到lg n次。进一步，习题79说明一个中数图最多有$O(n \log n)$条边。因此，如果不考虑步骤H3中的位设置操作，算法总的运行时间是$O(n (\log n)^2)$。

读者可能想对表2中的中数图手工执行算法H，该中数图的顶点代表4个变量$\{w, x, y, z\}$上的12个单调自对偶布尔函数，如（64）。所有实际涉及全部4个变量的此类函数都可以表示为5个元素的中数。从顶点$a=w$出发，算法计算边r_j—s_j的master表和表中所示的二进制标记。（实际处理顺序依赖于邻接表中顶点出现的次序，但最终标记除了列的排列外，次序都是一样的。）

表2 4个生成器的自由中数代数的标记



j	r_j	s_j	$l(s_j)$
		w	0000000
1	w	$\langle wwxyz\rangle$	0000001
2	$\langle wwxyz\rangle$	$\langle wyz\rangle$	0010001
3	$\langle wyz\rangle$	$\langle wxyzz\rangle$	0010101
4	$\langle wxyzz\rangle$	$\langle xyz\rangle$	0010111
5	$\langle wxyzz\rangle$	z	1010101
6	$\langle wyz\rangle$	$\langle wxyyz\rangle$	0010011
7	$\langle wxyyz\rangle$	y	0110011
8	$\langle wwxyz\rangle$	$\langle wxz\rangle$	0000101
9	$\langle wxz\rangle$	$\langle wxxyz\rangle$	0000111
10	$\langle wxxyz\rangle$	x	0001111
11	$\langle wwxyz\rangle$	$\langle wxy\rangle$	0000011

注意，每个标记$l(v)$中1的数目是v和开始顶点a间的距离。事实上，因为可从任何顶点出发，因此标记的唯一性告诉我们任何两个顶点的距离是顶点标记中比特位互不相同的位置的数目。

对于表2中的特殊中数图，由于其标记和相应的函数的真值表本质上是一样的，因此实际上可以不使用算法H，而以完全不同的方式进行处理。理由如下：我们可以说，简单函数w、x、y和z的真值表分别为$t(w)=0000000011111111$、$t(x)=0000111100001111$、$t(y)=0011001100110011$和$t(z)=0101010101010101$，因此$\langle wwxyz\rangle$的真值表是按位众数函数$\langle t(w)t(w)t(x)t(y)t(z)\rangle$，值就是二进制串0000000101111111。通过类似计算我们可以得出所有其他顶点的真值表。

任意自对偶函数的真值表的后半部分和前半部分除了互补、互逆外是一样的，

因此可以略去。进一步，4个真值表的最左位总是0。表2显示了剩下的7位，独有的特性保证了算法H在处理所给定的图时，除了列的排列可能不同外，产生的结果是相同的。

上述推理告诉我们，表2中图的边对应其真值表几乎是一样的一对函数。通过转变对应真值表的两个互补位，我们就可以在相邻顶点间移动。事实上，每个顶点的度恰好是由该顶点所表示的单调自对偶函数的析取式中基本蕴涵项的数目（参见习题70和84）。

***中数集** 中数集（median set）是二进制向量的汇集，有性质若$x\in X$，$y\in X$且$z\in X$，则$\langle xyz\rangle\in X$。如计算中数标记般，这里的中数是按成分值进行计算的。Thomas Schaefer在1978年注意到，中数集给出了一个刻画定理H中Horn函数特性的富有吸引力的对应物。

定理S 布尔函数$f(x_1, \cdots, x_n)$可表示为Krom子句的合取，当且仅当对所有布尔值x_j、y_j和z_j

$$f(x_1, \cdots, x_n)=f(y_1, \cdots, y_n)=f(z_1, \cdots, z_n)=1$$

蕴涵

$$f(\langle x_1y_1z_1\rangle, \cdots, \langle x_ny_nz_n\rangle)=1 \tag{68}$$

证明：[《STOC》10(1978), 216—226，引理3.1B]。如果有$x_1\vee x_2=y_1\vee y_2=z_1\vee z_2=1$，不妨假设$x_1\leqslant y_1\leqslant z_1$，则由于$y_1=0$蕴涵$x_2=y_2=1$，因此$\langle x_1y_1z_1\rangle\vee\langle x_2y_2z_2\rangle=y_1\vee\langle x_2y_2z_2\rangle=1$。由此（68）是必要条件。

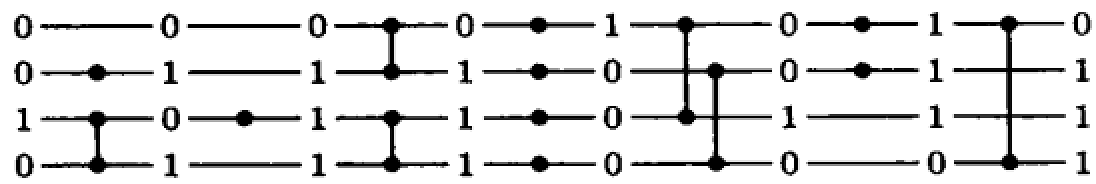
反之，如果（68）成立，令$u_1\vee\cdots\vee u_k$是f的一个基本子句，其中每个u_j是文字。则对于$1\leqslant j\leqslant k$，子句$u_1\vee\cdots\vee u_{j-1}\vee u_{j+1}\vee\cdots\vee u_k$不是$f$的一个子句，因此存在向量$\boldsymbol{x}^{(j)}$满足$(f)\boldsymbol{x}^{(j)}=1$，但对所有$i\neq j$有$u_i^{(j)}=0$。如果$k\geqslant 3$，对所有$1\leqslant i\leqslant k$，中数$<\boldsymbol{x}^{(1)}\boldsymbol{x}^{(2)}\boldsymbol{x}^{(3)}>$有$u_i=0$；但因为已经假定$u_1\vee\cdots\vee u_k$是一个子句，这是不可能的。因此$k\leqslant 2$。▌

所以中数集与“2SAT实例”满足某2CNF公式f的点集是一样的。

一个中数集称为是精简的（reduced），如果其向量$\boldsymbol{x}=\boldsymbol{x}_1\cdots\boldsymbol{x}_t$不含冗余成分。换句话说，对每个坐标位置$k$，一个精简中数集中至少有两个向量$\boldsymbol{x}^{(k)}$和$\boldsymbol{y}^{(k)}$，满足对所有$i\neq k$，$\boldsymbol{x}_k^{(k)}=0$且$\boldsymbol{y}_k^{(k)}=1$但$\boldsymbol{x}_i^{(k)}=\boldsymbol{y}_i^{(k)}$。我们已经看到一个中数图的标记满足该条件，事实上，如果坐标k对应于图中的边u—v，我们可以令$\boldsymbol{x}^{(k)}$和$\boldsymbol{y}^{(k)}$为u和v的标记。反之，任意的精简中数集X也定义了一个中数图，图中顶点对应X中的每个元素，顶点的邻接性则由坐标是否除了一位其他量位相等所定义。因为我们知道中数标记本质上是独一无二的，因此这些顶点的中数标记必等于原来X中的向量。

中数标记和精简中数集也可由另一种富有启发的方式刻画，该方式是5.3.4节所研究的比较器模块（comparator module）网络。注意一下，该节提到这种网络对“显然有序”的数进行排序是有用的，定理5.3.4Z说明一个比较器网络可对所有$n!$种可能的输入进行排序，当且仅当其可正确地对所有由0和1所组成的2^n种组合进行正确排序。当一个比较器模块和两条水平线相连，从左边输入x和y，右边输

出的是与输入相同的两个值，上一条线输出$\min(x, y)=x\wedge y$，下一条线输出$\max(x, y)=x\vee y$。将概念稍微扩展一下，现在允许反转器模块（inverter module），其将0改为1或反之。例如，下面就是一个比较–反转器网络（简称为CI网），其将二进制值0010转换为0111。



(69)

（图中单个的黑点表示反转器）实际上，该网络进行的转换是：

$0000 \mapsto 0110$	$0100 \mapsto 0111$	$1000 \mapsto 0111$	$1100 \mapsto 0110$
$0001 \mapsto 0111$	$0101 \mapsto 1111$	$1001 \mapsto 0101$	$1101 \mapsto 0111$
$0010 \mapsto 0111$	$0110 \mapsto 1111$	$1010 \mapsto 0101$	$1110 \mapsto 0111$
$0011 \mapsto 0110$	$0111 \mapsto 0111$	$1011 \mapsto 0111$	$1111 \mapsto 0110$

(70)

假定一个CI网将位串$x=x_1\cdots x_t$转换为位串$x_1'\cdots x_t'=f(x)$。这里的函数f将t立方体映射为自身，事实上是一同态映射。换句话说，当t立方体中有x—y时，$f(x)$—$f(y)$：因为网络中的每个模块都有这样的行为，因此改变x中的一位总是恰好引起$f(x)$中的一位发生变化。而且，CI网和中数标记有着重要的联系。

定理F *每个t位中数标记集X都可表示为一个计算布尔函数$f(x)$的比较–反转器网络，该网络具有这样的性质：对所有位向量$x_1\cdots x_t$，$f(x)\in X$，同时对所有$x\in X$有$f(x)=x$。*

证明：[Tomás Feder，《Memoirs Amer. Math. Sco.》**555**（1995），1—233，引理3.37；也可参见D. H. Wiedemann的博士论文（University of Waterloo，1986）]考虑中数标记中第i列和第j列，其中$1\leqslant i<j\leqslant t$。观察整个标记集合，因为中数标记不含冗余列，因此任何这样的列对至少包含4种可能性{00，01，10，11}中的3种。如果两列中缺少值00，01，10或11中的一个就写成$\bar{j}\to i$，$j\to i$，$i\to j$或$i\to\bar{j}$；我们也可分别记其等价关系为$\bar{i}\to j$，$\bar{i}\to\bar{j}$，$\bar{j}\to\bar{i}$或$j\to\bar{i}$，后者涉及$\bar{i}$而非i。例如，表2中的标记给出了如下关系：

$$\begin{array}{ll}
1\to\bar{2},3,\bar{4},5,\bar{6},7 & 2,\bar{3},4,\bar{5},6,\bar{7}\to\bar{1}\\
2\to 3,\bar{4},\bar{5},6,7 & \bar{3},4,5,\bar{6},\bar{7}\to\bar{2}\\
3\to\bar{4},7 & 4,\bar{7}\to\bar{3}\\
4\to 5,6,7 & \bar{5},\bar{6},\bar{7}\to\bar{4}\\
5\to 7 & \bar{7}\to\bar{5}\\
6\to 7 & \bar{7}\to\bar{6}
\end{array}\qquad(71)$$

（因为所有4种可能性都在这些列中出现，因此3和5间没有关系。但11不在第3列和第4列中出现，因此$3\to\bar{4}$。标记的第3列有一个1的顶点是那些更接近$\langle wyz\rangle$而非$\langle wwxyz\rangle$的顶点，因为这些顶点也更接近$\langle wxxyz\rangle$而非x，于是它们形成了一个凸集，其标记的第4列总是0。）

文字$\{1, \bar{1}, 2, \bar{2}, \cdots, t, \bar{t}\}$间的这些关系不含回路，因此总可以依拓扑序排成反对称序列$u_1u_2\cdots u_{2t}$，其中u_j是u_{2t+1-j}的补。例如：

$$1\ \bar{7}\ 4\ 2\ \bar{3}\ \bar{5}\ \bar{6}\ 6\ 5\ 3\ \bar{2}\ \bar{4}\ 7\ \bar{1} \tag{72}$$

是(71)中的关系拓扑排序的一种结果。

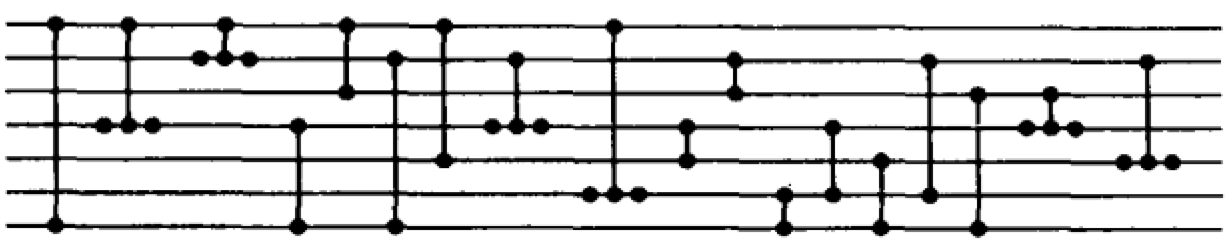
现在着手构造网络，从t条空线出发，接着对$d=2t-2$，$2t-3$，…，1（以此顺序）和$k=1$，2，…，$t-\lceil d/2\rceil$检测拓扑序中的元素u_k和u_{k+d}。如果$u_k \leqslant u_{k+d}$是列i和列j间的关系，其中$i<j$，则将附加新模块到网络中的线i和线j上，如下所示。

如果 $i\to j$　　如果 $i\to\bar{j}$　　如果 $\bar{\imath}\to j$　　如果 $\bar{\imath}\to\bar{j}$



(73)

例如，从(71)和(72)我们先执行$1\to7$，接着$1\to\bar{4}$，接着$1\to\bar{2}$，接着$\bar{7}\to\bar{4}$（亦即$4\to7$）等，可得如下网络：



(74)

（这好像是围棋棋谱图。但，比如说，当u_k是$\bar{7}$而u_{k+d}是3时，就没有添加新模块了，原因是$\bar{3}\to7$不在(71)中出现。）

习题89证明了(73)中每个新的模块组都保持先前所有的不等式并形成一个新的不等式。因此，如果x是任意的输入变量，而$f(x)$满足所有的不等式，则由定理S知，$f(x)\in X$。反之，如果$x\in X$，网络中的每个模块组都不改变x。▮

推论F 假定定理T中的中数标记对于按位AND操作和按位OR操作是闭的，即当$x\in X$和$y\in X$时，$x\&y\in X$且$x|y\in X$，则存在坐标的一种排列，在该排列下，标记可由仅含比较器模块的网络表示。

证明：所有标记的按位AND的结果是0…0，而按位OR的结果是1…1。因此列之间仅有的可能关系是$i\to j$和$j\to i$。经拓扑排序并对列重命名，可以保证当$i<j$时只有$i\to j$这种情景出现；在这种情况下，证明中的网络构造就不使用反转器了。▮

一般地，如果G是一个任意的图，f是将G的顶点映射到G的顶点子集X的同态映射，如果对所有$x\in X$有$f(x)=x$，则称f为回缩（retraction）；如果存在这样的f，则称X是G的一个回缩（retract）。该概念在图论中的重要性首先由Pavol Hell指出[参见《Lecture Notes in Math.》406（1974），291－301]。例如，一个结论是，X中顶点间的距离，即最短路径上边的数目，就算路径限制于X，结果仍是相同的（参见习题93）。

定理F表明中数标记的每个t维集是t维超立方体的一个回缩。反之，习题94则表明超立方体的回缩结果总是中数图。

阈函数（Threshold function）一种特别吸引人而又重要的布尔函数$f(x_1, x_2,\cdots, x_n)$，是由下述公式

$$f(x_1, x_2, \cdots, x_n)=[w_1x_1+w_2x_2+\cdots+w_nx_n \geqslant t] \tag{75}$$

定义的，其中$w_1w_2\cdots w_n$是整数“权重”，t是整数阈值。就算所有的权重都是单位值，

阈函数也很重要，例如：

$$x_1 \wedge x_2 \wedge \cdots \wedge x_n = [x_1 + x_2 + \cdots + x_n \geqslant n] \tag{76}$$

$$x_1 \vee x_2 \vee \cdots \vee x_n = [x_1 + x_2 + \cdots + x_n \geqslant 1] \tag{77}$$

和

$$\langle x_1 x_2 \cdots x_{2t-1} \rangle = [x_1 + x_2 + \cdots + x_{2t-1} \geqslant t] \tag{78}$$

其中，$\langle x_1, x_2, \cdots, x_{2t-1}\rangle$表示由奇数个布尔值$\{x_1, x_2, \cdots, x_{2t-1}\}$组成的多重集的中值（或众值）。特别地，基本映射$x \wedge y$、$x \vee y$和$\langle xyz\rangle$都是阈函数，而

$$\bar{x} = [-x \geqslant 0] \tag{79}$$

也是阈函数。

使用更一般的权重可以得到许多有意义的函数，如

$$[2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + x_n \geqslant (t_1 t_2 \cdots t_n)_2] \tag{80}$$

它为真当且仅当二进制串$x_1x_2\cdots x_n$就字典序而言大于等于给定的二进制串$t_1t_2\cdots t_n$。给定一个规模大小分别为$w_1, w_2, \cdots, w_n$的n个物体，问是否存在这些物体的一个子集，它恰好可放入大小为$t-1$的背包，其对应的就是求证$f(x_1, x_2, \cdots, x_n)=0$是否可能的问题，其中$x_j=0$表示物体$j$在子集中出现。而基于最初由W. McCullough和W. Pitts在《Bull. Math. Biophysics》5（1943），115－133一文所提出的神经元简单模型，已经有上千篇基于阈函数的“神经元网络”研究文章发表。

令$x_j \leftarrow \bar{x}_j$，$w_j \leftarrow -w_j$，$t \leftarrow t+|w_j|$可摆脱负数权重。因此一个一般的阈函数可以约简为一个正阈函数，其中所有的权重皆为正数。进一步，任意的正阈函数(75)可以表示为特殊的奇中值/众值函数，原因是

$$\langle 0^a 1^b x_1^{w_1} x_2^{w_2} \cdots x_n^{w_n} \rangle = [b + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \geqslant b + t] \tag{81}$$

其中x^m表示x的m个复制，而a和b由下面规则定义

$$a = \max(0,\ 2t-1-w), \quad b = \max(0,\ w+1-2t), \quad w = w_1 + w_2 + \cdots + w_n \tag{82}$$

例如，当所有权重都为1时，我们有

$$\langle 0^{n-1} x_1 \cdots x_n \rangle = x_1 \wedge \cdots \wedge x_n,\ \langle 1^{n-1} x_1 \cdots x_n \rangle = x_1 \vee \cdots \vee x_n \tag{83}$$

对于$n=2$，我们已经在(45)中看到了这些公式。一般地，或者a或者b为0，此时等式(81)的左边指定了$2T-1$个元素的中值，其中

$$T = b + t = \max(t,\ w_1 + w_2 + \cdots + w_n + 1 - t) \tag{84}$$

由于众数函数（majority function）显然满足取消律（cancellation law）

$$\langle 01 x_1 x_2 \cdots x_{2t-1} \rangle = \langle x_1 x_2 \cdots x_{2t-1} \rangle \tag{85}$$

因此无法令a和b都大于0。

等式（81）的一个重要结论是令x_0为0或1，可得每个正阈函数都来自纯众数函数

$$g(x_0,\ x_1,\ x_2,\ \cdots,\ x_n) = \langle x_0^{a+b} x_1^{w_1} x_2^{w_2} \cdots x_n^{w_n} \rangle \tag{86}$$

换句话说，我们知道所有n个变量的阈函数，当且仅当我们知道所有不同的有$n+1$个或更少变量（不含常量）的奇中数函数。每个纯众数函数是单调且自对偶的，

因此，我们就看到了表2s_j列中4个变量$\{w,x,y,z\}$的众数函数，即$\langle w\rangle$、$\langle ww xyz\rangle$、$\langle wyz\rangle$、$\langle wxyzz\rangle$、$\langle xyz\rangle$、$\langle z\rangle$、$\langle wxyyz\rangle$、$\langle y\rangle$、$\langle wxz\rangle$、$\langle wxxyz\rangle$、$\langle x\rangle$和$\langle wxy\rangle$。通过令w=0或1，可得如下有3个变量的所有正阈函数$f(x,y,z)$。

$$\begin{gathered}\langle 0\rangle, \langle 1\rangle, \langle 00xyz\rangle, \langle 11xyz\rangle, \langle 0yz\rangle, \langle 1yz\rangle, \langle 0xyzz\rangle, \langle 1xyzz\rangle, \langle xyz\rangle, \langle z\rangle,\\ \langle 0xyyz\rangle, \langle 1xyyz\rangle, \langle y\rangle, \langle 0xz\rangle, \langle 1xz\rangle, \langle 0xxyz\rangle, \langle 1xxyz\rangle, \langle x\rangle, \langle 0xy\rangle, \langle 1xy\rangle\end{gathered} \quad (87)$$

所有150个4变量正阈函数都可由相似的方式从习题84答案中的自对偶众数函数得到。

有无穷多个权重序列（$w_1, w_2, \cdots, w_n$），但对给定的n，只有有限多个阈函数。由此可见，许多不同的权重序列显然是等价的。例如，考虑下面的纯众数函数

$$\langle x_1^2 x_2^3 x_3^5 x_4^7 x_5^{11} x_6^{13}\rangle$$

其中的权重为素数，通过对2^6种情况进行暴力式检查，可得

$$\langle x_1^2 x_2^3 x_3^5 x_4^7 x_5^{11} x_6^{13}\rangle = \langle x_1 x_2^2 x_3^2 x_4^3 x_5^4 x_6^5\rangle \quad (88)$$

由此可知，我们可以使用本质上较小的权重来表示同一函数。类似地，阈函数

$$[(x_1 x_2 \cdots x_{20})_2 \geqslant (01100100100001111110)_2] = \langle 1^{225028} x_1^{524288} x_2^{262144} \cdots x_{20}\rangle$$

是(80)的一个特例，可以简化为

$$\langle 1^{323} x_1^{764} x_2^{323} x_3^{323} x_4^{118} x_5^{118} x_6^{87} x_7^{31} x_8^{31} x_9^{25} x_{10}^{6} x_{11}^{6} x_{12}^{6} x_{13}^{6} x_{14} x_{15} x_{16} x_{17} x_{18} x_{19}\rangle \quad (89)$$

习题103说明了使用线性规划而非采取高强度暴力搜索寻找权重最小集的方法。

C. K. Chow发明了一种优雅的对任意阈函数赋以唯一标识的索引方案[《FOCS》2(1961)，34—38]。给定任意布尔函数$f(x_1, \cdots, x_n)$，令$N(f)$是使$f(x)=1$的向量$\boldsymbol{x}=(x_1,\cdots,x_n)$的数目，令$\Sigma(f)$是所有这些向量的和。例如，如果$f(x_1, x_2)=x_1\vee x_2$，则$N(f)=3$，而$\Sigma(f)=(0,1)+(1,0)+(1,1)=(2,2)$。

定理T 令$f(x_1, \cdots, x_n)$和$g(x_1, \cdots, x_n)$是布尔函数，满足$N(f)=N(g)$且$\Sigma(f)=\Sigma(g)$，其中f是阈函数，则有$f=g$。

证明：假定恰好有k个向量$x^{(1)}, \cdots, x^{(k)}$使$f(x^{(j)})=1$且$g(x^{(j)})=0$。因为$N(f)=N(g)$，其必恰好有k个向量$y^{(1)}, \cdots, y^{(k)}$满足$f(y^{(j)})=0$且$g(y^{(j)})=1$。因为$\Sigma(f)=\Sigma(g)$，故必有$x^{(1)}+\cdots+x^{(k)}=y^{(1)}+\cdots+y^{(k)}$。

现假定f是阈函数（75），则对$1\leqslant j\leqslant k$有$w\cdot x^{(j)}\geqslant t$且$w\cdot y^{(j)}<t$。然而，若$f\neq g$，有$k>0$且$w\cdot(x^{(1)}+\cdots+x^{(k)})\geqslant w\cdot(y^{(1)}+\cdots+y^{(k)})$，矛盾。■

阈函数有许多奇特的性质，其中一些出现在后面的习题中。阈函数经典理论由Saburo Muroga归纳并发表在其著作《阈逻辑及其应用》（《Threshold Logic and its Applications》，Wiley，1971）中。

对称布尔函数 称函数$f(x_1, \cdots, x_n)$是对称的，若$f(x_1, \cdots, x_n)$等于$f(x_{p(1)}, \cdots, x_{p(n)})$，其中$p(1)\cdots p(n)$是$\{1, \cdots, n\}$的所有排列。当所有的$x_j$皆为0或1时，该条件等价于$f$仅依赖于在参数中出现的1的数目，即“横向和”（sideways sum）$\nu x=\nu(x_1, \cdots, x_n)=x_1+\cdots+x_n$。通常使用$S_{k_1,k_2,\cdots,k_r}(x_1, \cdots, x_n)$这样的记法表示一个布尔函数其为真当且仅当$\nu x$的值或为$k_1$或为$k_2\cdots$或为$k_n$。例如，$S_{1,3,5}(v, w, x, y, z)=v\oplus w\oplus x\oplus$

$y\oplus z$, $S_{3,4,5}(v,w,x,y,z)=\langle vwxyz\rangle$, $S_{4,5}(v,w,x,y,z)=\langle 00vwxyz\rangle$。

许多对称应用涉及基本函数$S_k(x_1,\cdots,x_n)$，其仅当$\nu x=k$时值为真。例如，$S_3(x_1, x_2, x_3, x_4, x_5, x_6)$为真当且仅当参数$\{x_1, x_2, x_3, x_4, x_5, x_6\}$中恰好一半为真而另一半为假时。在这种情况下，显然有

$$S_k(x_1,\cdots,x_n)=S_{\geqslant k}(x_1,\cdots,x_n)\wedge\overline{S_{\geqslant k+1}(x_1,\cdots,x_n)} \tag{90}$$

其中$S_{\geqslant k}(x_1,\cdots,x_n)$是$S_{k,k+1,\cdots,k+n}(x_1,\cdots,x_n)$的缩写。而函数$S_{\geqslant k}(x_1,\cdots,x_n)$当然就是前面已经研究过的阈函数$[x_1+\cdots+x_n\geqslant k]$。

更加复杂的情况可以被处理为阈函数的阈函数。例如，我们有

$$\begin{aligned}s_{2,3,6,8,9}(x_1,\cdots,x_{12})&=[\nu x\geqslant 2+4[\nu x\geqslant 4]+2[\nu x\geqslant 7]+5[\nu x\geqslant 10]]\\&=\langle 00x_1\cdots x_{12}\langle 0^5\bar{x}_1\cdots\bar{x}_{12}\rangle^4\langle 1\bar{x}_1\cdots\bar{x}_{12}\rangle^2\langle 1^7\bar{x}_1\cdots\bar{x}12\rangle^5\rangle\end{aligned} \tag{91}$$

因为当$x_1+\cdots+x_{12}=(0，1，\cdots，12)$时，最外的25的众数中1的数目分别是（11，12，13，14，11，12，13，12，13，14，10，11，12）。通常，使用类似的两层方案也是可行的[R. C. Minnick，《IRE Trans》. EC-10(1961), 6—16]，但使用3层或更多层的逻辑方案可以将阈操作的数量进一步减少（参见习题113）。

已经发明了一些独特的求解对称布尔函数的技巧。例如，S. Muroga将下面这一非凡的公式序列归因于F. Sasaki。

$$\begin{aligned}&x_0\oplus x_1\oplus\cdots\oplus x_{2m}=\langle\bar{x}_0 s_1 s_2\cdots s_{2m}\rangle,\\&\qquad\text{当 } s_j=\langle x_0 x_j x_{j+1}\cdots x_{j+m-1}\bar{x}_{j+m}\bar{x}_{j+m+1}\cdots\bar{x}_{j+2m-1}\rangle\text{ 时}\end{aligned} \tag{92}$$

假定$m>0$且对于$k>0$，x_{2m+k}和x_k是一样的。特别地，当$m=1$和$m=2$时，我们有恒等式

$$x_0\oplus x_1\oplus x_2=\langle\bar{x}_0\langle x_0x_1\bar{x}_2\rangle\langle x_0x_2\bar{x}_1\rangle\rangle \tag{93}$$

$$x_0\oplus\cdots\oplus x_4=\langle\bar{x}_0\langle x_0x_1x_2\bar{x}_3\bar{x}_4\rangle\langle x_0x_2x_3\bar{x}_4\bar{x}_1\rangle\langle x_0x_3x_4\bar{x}_1\bar{x}_2\rangle\langle x_0x_4x_1\bar{x}_2\bar{x}_3\rangle\rangle \tag{94}$$

等式的右边是完全对称的，但并非显而易见（参见习题115）！

渠限化函数（canalizing function） 对于布尔函数$f(x_1,\cdots,x_n)$，如果通过检查最多一个变量的值就可以推出其值的话，就称该布尔函数是渠限化的（canalizing）或者“逼迫”（forcing）的。更精确地说，如果$n=0$或者存在下标j，当令$x_j=0$时，f有常量值或令$x_j=1$，f有常量值，则f是渠限化函数。例如，$f(x,y,z)=(x\oplus z)\vee\bar{y}$是渠限化函数，原因是当$y=0$时其值总为1（当$y=1$时，若不检查$x$和$z$的值我们就不知道$f$的值，但半块面包总比没有面包好）。此类函数，由Stuart Kauffman引入[《Lectures on mathematics in the Life Sciences》3(1972), 63—116；《J. Theoretical Biology》44(1974), 167—190]，在许多应用中，特别是在化学和生物学中，被证明是重要的。它们的一些性质在习题125～129中验证。

量化的考虑 我们已经研究过许多不同种类的布尔函数，因此自然就会有人问：有n个变量的每种类型的函数实际上有多少个？

表3列出了所有的函数。对于每个n，由于有2^{2^n}种可能的真值表，因此有2^{2^n}种可能性。这些函数中的一些是自对偶的，一些是单调的，一些满足定理P，即既是单调又是自对偶的。一些满足定理H，是Horn函数，一些满足定理S，是Krom函数，等等。

在表4中，两个函数若仅仅只是变量名不同，则认为是相等的。当n=2时，仅有12种不同的情况，因为（例如）$x\vee\bar{y}$和$\bar{x}\vee y$本质上是一样的。

表5更进一步，允许我们对单一变量求补，甚至本质不改变而对整个函数求补。在这种视角下，(x, y, z)上的256个布尔函数可分成14个不同的等价类。

代表	类大小	代表	类大小
0	2	$x\wedge(y\oplus z)$	24
x	6	$x\oplus(y\wedge z)$	24
$x\wedge y$	24	$(x\wedge y)\vee(\bar{x}\wedge z)$	24
$x\oplus y$	6	$(x\vee y)\wedge(x\oplus z)$	48
$x\wedge y\wedge z$	16	$(x\oplus y)\vee(x\oplus z)$	8
$x\oplus y\oplus z$	2	$\langle xyz\rangle$	8
$x\wedge(y\vee z)$	48	$S_1(x,y,z)$	16

(95)

7.2.3节将研究如何列出非等价的组合对象并对其进行计数。

表3 n个变量的布尔函数

	n=0	n=1	n=2	n=3	n=4	n=5	n=6
任意	2	4	16	256	65 536	4 294 967 296	18 446 744 073 709 551 616
自对偶	0	2	4	16	256	65 536	4 294 967 296
单调	2	3	6	20	168	7 581	7 828 354
两者皆是	0	1	2	4	12	81	2 646
Horn	2	4	14	122	4 960	2 771 104	151 947 502 948
Krom	2	4	16	166	4 170	224 716	24 445 368
阈	2	4	14	104	1 882	94 572	15 028 134
对称	2	4	8	16	32	64	128
渠限化	2	4	14	120	3 514	1 292 276	103 071 426 294

表4 关于变量排列的不同布尔函数

	n=0	n=1	n=2	n=3	n=4	n=5	n=6
任意	2	4	12	80	3 984	37 333 248	25 626 412 338 274 304
自对偶	0	2	2	8	32	1 088	6 385 408
单调	2	3	5	10	30	210	16 353
两者皆是	0	1	1	2	3	7	30
Horn	2	4	10	38	368	29 328	216 591 692
Krom	2	4	12	48	308	3 028	49 490
阈	2	4	10	34	178	1 720	590 440
渠限化	2	4	10	38	294	15 774	149 325 022

表5 关于变量补/排列的不同布尔函数

	n=0	n=1	n=2	n=3	n=4	n=5	n=6
任意	1	2	4	14	222	616 126	200 253 952 527 184
自对偶	0	1	1	3	7	83	109 950
阈	1	2	3	6	15	63	567
两者皆是	0	1	1	2	3	7	21
渠限化	1	2	3	6	22	402	1 228 158

习　题

1. [15]（Lewis Carroll）请解释一下本节开头部分特维德里迪的观点。[提示：参考表1。]

2. [17]在遥远的平卡斯星球上，逻辑学家用符号1代表“假”而用0来代表“真”。举个例子来说，他们称作“or”的二元运算与我们用的∧一样有着如下性质：

$$1 \text{ or } 1=1,\quad 1 \text{ or } 0=0,\quad 0 \text{ or } 1=0,\quad 0 \text{ or } 0=0$$

请问与平卡斯人称作“假”，“并且”，…，“与非”，“有效”（见表1）的16种逻辑运算联系在一起的运算分别是什么？

▸3. [13]假设用-1代表逻辑值假，+1代表逻辑值真，而不是用0和1。那么表1中那个运算∘对应于(a) $\max(x, y)$? (b)$\min(x, y)$? (c)$-x$? (d)$x \cdot y$?

4. [24]（H. M. Sheffer）这道习题的目的是为了说明表1中的所有运算都能用NAND予以表示。(a)对表中的每一个运算∘，给出等价于$x \circ y$且只使用运算符 $\barwedge$ 的表达式。给出的表达式要尽量短。比如，运算符 ⊥ 的答案就只是简单的“x”，而⊏的答案是“$x \barwedge x$”。请不要在你的表达式中使用常量0或者1；(b)类似地，请给出16个可包含常量的短表达式，比如x⊏y现在可以表示为“$x \barwedge 1$”。

5. [24]考虑将第4题中的基本运算 $\barwedge$ 换为 ⊄。

6. [21]（E. Schröder）(a)表1中的16种运算哪些满足结合律，换句话说，它们中哪些有性质$x \circ (y \circ z)=(x \circ y) \circ z$？(b)哪些有性质$(x \circ y) \circ (y \circ z)=x \circ z$？

7. [20]表1中的哪些运算当且仅当$y \circ z=x$才有$x \circ y=z$？

8. [24] 16^2对运算(∘，□)中有哪些满足左分配律$x \circ (y \square z)=(x \circ y) \square (x \circ z)$？

9. [16]判断以下式子的对错。

(a) $(x \oplus y) \vee z=(x \vee z) \oplus (y \vee z)$；(b) $(w \oplus x \oplus y) \vee z=(w \vee z) \oplus (x \vee z) \oplus (y \vee z)$；

(c) $(x \oplus y) \vee (y \oplus z)=(x \oplus z) \vee (y \oplus z)$。

10. [17]随机函数(22)的多重线性表示是什么？

11. [M25]是否存在一种直观方式，借此可正确理解$f(x_1, \cdots, x_n)$的多重线性表示何时包含某一个项，比如说$x_2x_3x_6x_8$？（参见（19）。）

▸12. [M23]一个布尔函数的整系数多重线性（integer multilinear representation）表示是将一个像（19）那样的表达式扩展为一个带整系数的多项式$f(x_1, \cdots, x_n)$。以这种方式表示的$f(x_1, \cdots, x_n)$不用对2取模，对所有2^n种可能的0−1向量$(x_1, \cdots, x_n)$均具有正确的值（0或1）。比如，与（19）对应的整系数多重线性表示是$1-xy-xz-yz+3xyz$。

a) “随机”函数（22）的整系数多重线性表示是什么？

b) 在$f(x_1, \cdots, x_n)$这样的表达式中，系数最大能为多少？

c) 证明：在每个整系数多重线性表示中，只要$x_1, \cdots, x_n$是满足$0 \leqslant x_1, \cdots, x_n \leqslant 1$的实数，则有$0 \leqslant f(x_1, \cdots, x_n) \leqslant 1$。

d) 类似，如果当$\{x_1, \cdots, x_n\} \subseteq \{0, 1\}$时有$f(x_1, \cdots, x_n) \leqslant g(x_1, \cdots, x_n)$，证明当$\{x_1, \cdots, x_n\} \subseteq [0, 1]$时有$f(x_1, \cdots, x_n) \leqslant g(x_1, \cdots, x_n)$。

e) 如果f单调且对于$1\leqslant j\leqslant n$有$0\leqslant x_j\leqslant y_j\leqslant 1$，证明$f(x)\leqslant f(y)$。

▶ 13. [20]考虑一个由n个部件组成的系统，它们中的每一个可能处在“工作”状态或者是“失效”状态。如果x_j表示“第j个部件正在工作”，则布尔函数$x_1\wedge(\bar{x}_2\vee\bar{x}_3)$表示“第1个部件正在工作，而部件2或者部件3处于失效状态”；并且$S_3(x_1,\cdots,x_n)$表示“3个部件都处在工作状态”。

假设每个部件j处在正常工作状态的概率为p_j，并与其他部件不相关。证明布尔函数$f(x_1,\cdots,x_n)$为真的概率为$F(p_1,\cdots,p_n)$，其中F是个由变量$p_1,\cdots,p_n$组成的多项式。

14. [20]第13题中的概率函数$F(p_1,\cdots,p_n)$通常被称为系统的可用性(availability)。试找出当概率(p_1,p_2,p_3)是以下三种情况时，具有最大可用性的自对偶函数$f(x_1,\cdots,x_n)$：(a) (.9,.8,.7)；(b) (.8,.6,.4)；(c) (.8,.6,.1)。

▶ 15. [M20]如果$f(x_1,\cdots,x_n)$是一个布尔函数，证明存在一个多项式$F(x)$，当x是整数时$F(x)$也是整数，而且$f(x_1,\cdots,x_n)=F((x_n\cdots x_1)_2)\bmod 2$。提示：考虑$\binom{x}{k}\bmod 2$。

16. [13]能否在一个满析取范式中用⊕代替每一个∨的出现？

17. [10]根据德摩根（De Morgan）定律，一个如（25）那样的一般析取范式不仅是AND的OR，也是NAND的NAND：

$$\overline{\overline{(u_{11}\wedge\cdots\wedge u_{1s_1})}\wedge\cdots\wedge\overline{(u_{m1}\wedge\cdots\wedge u_{ms_m})}}$$

因此可以认为这两个逻辑层是相同的。

一个叫J. H. Quick的学生将这个表达式重新写为如下形式。

$$(u_{11}\,\overline{\wedge}\cdots\overline{\wedge}\,u_{1s_1})\,\overline{\wedge}\cdots\overline{\wedge}\,(u_{m1}\,\overline{\wedge}\cdots\overline{\wedge}\,u_{ms_m})$$

这是正确的吗？

▶ 18. [20]令$u_1\wedge\cdots\wedge u_s$为布尔函数$f$的析取范式的一个蕴涵项，令$v_1\vee\cdots\vee v_t$为同一函数的合取范式的一个子句。证明，存在某个$i$和$j$，使得$u_i=v_j$。

19. [20]（22）中的“随机”函数的基本合取式是什么？

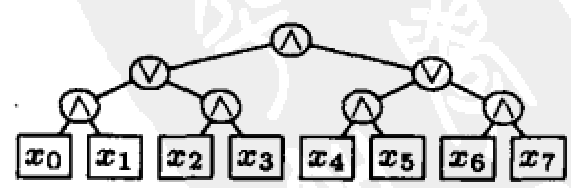
20. [M21]判断正误：$f\wedge g$的每一个基本蕴涵项都能写成$f'\wedge g'$，其中f'是f的基本蕴涵项，而g'是g的基本蕴涵项。

21. [M20]证明一个非常量布尔函数是单调的，当且仅当其可完全用运算符∧和∨表示。

22. [20]假设在（16）中$f(x_1,\cdots,x_n)=g(x_1,\cdots,x_{n-1})\oplus h(x_1,\cdots,x_{n-1})\wedge x_n$，$f$单调必须而且仅须$g$和$h$具有什么条件？

23. [15]$(v\wedge w\wedge x)\vee(v\wedge x\wedge z)\vee(x\wedge y\wedge z)$的基本合取式是什么？

24. [M20]考虑一个具有2^k个叶子的完全二叉树，这里显示的是$k=3$的情况。



在每层交替运行∧和或∨，在根上使用∧得到表达式如$((x_0\wedge x_1)\vee(x_2\wedge x_3))\vee((x_4\wedge x_5)\vee(x_6\wedge x_7))$。结果函数中包含多少个基本蕴涵项？

25. [M21] $(x_1\vee x_2)\wedge(x_2\vee x_3)\wedge\cdots\wedge(x_{n-1}\vee x_n)$有多少个基本蕴涵项?

26. [M23]令$\mathcal{F}$和$\mathcal{G}$为一个单调CNF和一个单调DNF的基本子句和基本蕴涵项的两族索引集：

$$f(x)=\bigwedge_{I\in\mathcal{F}}\bigvee_{i\in I}x_i;\quad g(x)=\bigvee_{J\in\mathcal{G}}\bigwedge_{j\in J}x_j$$

找出一个x，使得当以下任何一个条件成立时，有$f(x)\neq g(x)$。

a) 存在一个$I\in\mathcal{F}$和一个$J\in\mathcal{G}$使得$I\cap J=\varnothing$。

b) $\bigcup_{I\in\mathcal{F}}I\neq\bigcup_{J\in\mathcal{G}}J$。

c) 存在一个$I\in\mathcal{F}$使得$|I|>|\mathcal{G}|$，或者存在一个$J\in\mathcal{G}$使得$|J|>|\mathcal{F}|$。

d) $\sum_{I\in\mathcal{F}}2^{n-|I|}+\sum_{J\in\mathcal{G}}2^{n-|J|}<2^n$，其中$n=\left|\bigcup_{I\in\mathcal{F}}I\right|$。

27. [M31]接习题26，考虑如下算法$X(\mathcal{F},\mathcal{G})$，它或者返回满足$f(x)\neq g(x)$的向量$\boldsymbol{x}$，或者当$f=g$时返回$\boldsymbol{\Lambda}$。

X1. [检验必要条件。] 如果条件为26题中的(a)、(b)、(c)或(d)则返回一个恰当的$\boldsymbol{x}$值。

X2. [成立?] 如果$|\mathcal{F}||\mathcal{G}|\leqslant 1$，返回$\boldsymbol{\Lambda}$。

X3. [递归。] 对如下约简索引族进行计算，以得到“最好”的索引k：

$$\mathcal{F}_1=\{I|I\in\mathcal{F},k\notin J\},\ \mathcal{F}_0=\mathcal{F}_1\cup\{I|k\in I,I\cup\{k\}\in F\};$$
$$\mathcal{G}_0=\{J|J\in\mathcal{G},k\notin J\},\ \mathcal{G}_1=\mathcal{G}_0\cup\{J|k\notin J,J\cup\{k\}\in\mathcal{G}\}。$$

删除$\mathcal{F}_0$和$\mathcal{G}_l$中任何一个包含同一个集合中其他成员的成员。索引k应该使得比值$\rho=\min(|\mathcal{F}_1|/|\mathcal{F}|,|\mathcal{G}_0|/|\mathcal{G}|)$尽可能小。如果$X(\mathcal{F}_0,\mathcal{G}_0)$返回一个向量$\boldsymbol{x}$，则返回同一个由$x_k=0$扩展成的向量。否则如果$X(\mathcal{F}_1,\mathcal{G}_1)$返回一个向量$\boldsymbol{x}$，则返回同一个由$x_k=1$扩展成的向量。否则返回$\boldsymbol{\Lambda}$。▮

如果$N=|\mathcal{F}|+|\mathcal{G}|$，证明步骤X1最多运行$N^{o(\log N)^2}$次。提示：证明在步骤X3中总有$\rho\leqslant 1-1/\lg N$。

28. [21]（W. V. Quine，1952。）如果$f(x_1,\cdots,x_n)$是具有基本蕴涵项$p_1,\cdots,p_q$的布尔函数，令$g(y_1,\cdots,y_q)=\wedge_{f(x)=1}\vee\{y_j|p_j(x)=1\}$。比如，随机函数(22)在(28)的8个点上为真，而且它有由（29）和（30）给出的5个基本蕴涵项；所以，在这个例子中$g(y_1,\cdots,y_5)$是

$$(y_1\vee y_2)\wedge(y_1)\wedge(y_2\vee y_3)\wedge(y_4)\wedge(y_3\vee y_5)\wedge(y_5)\wedge(y_5)\wedge(y_4\vee y_5)$$
$$=(y_1\wedge y_2\wedge y_4\wedge y_5)\vee(y_1\wedge y_3\wedge y_4\wedge y_5)$$

证明f的每一个最短DNF表达式对应单调函数g的一个基本蕴涵项。

29. [22]（以下习题讨论一种处理布尔函数蕴涵项的算法，其将n维立方体的顶点表示为n位数$(b_{n-1}\cdots b_1b_0)_2$而非位串$x_1\cdots x_n$。）给定一个位的位置j，给定n位值$v_0<v_1<\cdots<v_{m-1}$，试解释如何按k递增的顺序找到所有的(k,k')，使得$0\leqslant k<k'<m$并且$v_k=v_k\oplus 2^j$。算法运行的时间必须为$O(m)$，假设在n比特字上进行按位操作的时间为常数。

▸30. [27]书中指出一个布尔函数的蕴涵项可以看作一个子立方体，如01*0*，包

含于对所有点都为真的函数集合V中。每一个子立方体可以表示为一对二进制数字$a=(a_{n-1}\cdots a_0)_2$和$b=(b_{n-1}\cdots b_0)_2$，其中a记录了星号的位置而b记录了非星号位置的比特位。比如，数字$a=(00101)_2$和$b=(00101)_2$表示子立方体$c=01*0*$。我们总有$a\&b=0$。

一个子立方体的"j-伙伴"（j-buddy）定义为当$a_j=0$时将b变为$b\oplus 2^j$。比如，01*0*有3个伙伴，也就是4－伙伴11*0*、3－伙伴00*0*0和1－伙伴01*1*。每一个子立方体$c\subseteq V$可以被指定一个标签$(t_{n-1}\cdots t_0)_2$，其中$t_j=1$当且仅当c的j-伙伴是有定义的而且包含于V。在这种定义下c代表一个极大子立方体（因此是一个基本蕴涵项）当且仅当它的标签值为0。

利用这些概念设计一种算法来找出一个给定的集合V的所有极大子立方体(a, b)，其中V由n位数字$v_0<v_1<\cdots<v_{m-1}$表示。

▸31. [28]当一个布尔函数为真时，习题30中的算法需要所有点的完整列表，而这个列表可能会非常长。因此除非必要，我们宁愿去直接计算子立方体，而不是深入到显式的n元组层次。这种高层次方法的关键是使用子立方体c和c'的一致集（consensus）$c\sqcup c'$，其为满足

$$c''\subseteq c\cup c',\ c''\not\subseteq c \text{且} c''\not\subseteq c'$$

的最大子立方体c''，这样的c''并不总是存在。比如，如果$c=000*$且$c'=*111$，每一个包含于$c\cup c'$的子立方体或者包含于c，或者包含于c'。

a) 证明当一致集存在时，对每一个坐标我们可以用以下公式计算它的分量：

$$x\sqcup x=x\sqcup *=*\sqcup x=x \text{且} x\sqcup \bar{x}=*\sqcup *=*，\text{对所有的}x=0\text{和}x=1。$$

进一步，$c\sqcup c'$存在当且仅当规则$x\sqcup\bar{x}=*$已经恰好被应用到其中的一个分量上。

b) 有k个星号的子立方体称为k-立方体。证明：如果c是一个k-立方体，c'是一个k'-立方体，如果一致集$c''=c\sqcup c'$存在，则c''是一个k''-立方体，其中$1\leqslant k''\leqslant \min(k, k')+1$。

c) 如果C和C'是子立方体的集合，令

$$C\sqcup C'=\{c\sqcup c' | c\in C, c'\in C', \text{且} c\sqcup c' \text{存在}\}$$

解释为什么下面的算法是正确的。

算法E（寻找极大子立方体） 给定一个n维立方体的子立方体的一个集合C，该算法输出$V=\bigcup_{c\in C} c$的极大子立方体而没有真正对集合V本身进行计算。

E1. [初始化。] 令$j\leftarrow 0$，如果C的一个子立方体c被另一个所包含，则删除它。

E2. [完成？] （此时，每一个包含于V的j-立方体包含于C的某个元素中，而且当$k<j$时，C中没有k-立方体）如果C是空集，则算法终止。

E3. [采取一致。] 令$C'\leftarrow C\sqcup C$，对于$k\leqslant j$，删除C'中的所有k-立方体的子立方体。当进行本次计算时，如果C的j-立方体c使得$c\sqcup C$不能产生一个C'的$(j+1)$-立方体，则输出c。

E4. [推进。] 令$C\leftarrow C\cup C'$，从并集里删除所有j-立方体，如果C的任一个子立方体c被另一个所包含，则删除。令$j\leftarrow j+1$并回到E2。▌

（参见7.1.3节的习题142中一个更有效的计算方法。）

▸ 32. [M29]令$c_1, \cdots, c_m$为n-立方体的子立方体。

a) 证明$c_1 \cup \cdots \cup c_m$至多包含一个极大子立方体c，使得c对所有的$j \in \{1, \cdots, m\}$都有c不包含于$c_1 \cup \cdots \cup c_{j-1} \cup c_{j+1} \cup \cdots \cup c_m$。（如果$c$存在，我们称它为$c_1, \cdots, c_m$的广义一致集（generalized consensus），当$m=2$时，按习题31的记法有$c=c_1 \sqcup c_2$。）

b) 寻找一个由m个子立方体构成的集合，使得$\{c_1, \cdots, c_m\}$的2^m-1个非空子集每个都有一个广义一致集。

c) 证明一个带有m个蕴涵项的DNF至多有2^m-1个基本蕴涵项。

d) 寻找一个带有m个蕴涵项和2^m-1个基本蕴涵项的DNF。

33. [M21]令$f(x_1, \cdots, x_n)$为$\binom{2^n}{m}$个布尔函数中的一个在m个点都为真的函数。如果f是随机选取的，那么在下列情况中，$x_1 \wedge \cdots \wedge x_k$各是什么概率。(a) f的一个蕴涵项？(b) f的一个基本蕴涵项？[以和的形式给出（b）的答案；但当$k=n$时对闭形式进行计算。]

▸ 34. [HM37]接习题33，令$c(m, n)$为蕴涵项数目的平均总数，又令$p(m, n)$为基本蕴涵项数目的平均总数。

a) 如果$0 \leqslant m \leqslant 2^n/n$，证明$m \leqslant c(m, n) \leqslant \frac{3}{2}m + O(m/n)$且$p(m, n) >= me^{-1} + O(m/n)$；从而在这个范围内有$p(m, n) = \Theta(c(m, n))$。

b) 现在令$2^n/n \leqslant m \leqslant (1-\varepsilon)2^n$，其中$\varepsilon$是一个固定的正常数。定义数字$t$和$\alpha_{mn}$的关系如下：

$$n^{-4/3} \leqslant \left(\frac{m}{2^n}\right)^{2^t} = \alpha_{mn} < n^{-2/3}，\ t\text{为整数}$$

请用n、t和α_{mn}来表示$c(m, n)$和$p(m, n)$的渐近值。

[提示：证明几乎所有的蕴涵项都恰有$n-t$或者$n-t-1$个文字。]

c) 估算$c(m, n)/p(m, n)$，当$m=2^{n-1}$且$n = \left\lfloor (\ln t - \ln\ln t)2^{2^t} \right\rfloor$，$t$为整数时。

d) 证明当$m \leqslant (1-\varepsilon)2^n$时，有$c(m, n)/p(m, n) = O(\log\log n/\log\log\log n)$。

▸ 35. [M25]一个DNF称为正交（orthogonal）的，如果它的蕴涵项对应于不相交的子立方体。当要计算或者估计习题13中的可靠性多项式时，正交析取范式是非常有用的。

每个函数的完全DNF显然都是正交的，因为它的子立方体都是单点的。但是我们经常能发现一个正交的DNF有着明显较少的蕴涵项，特别当函数是单调的时候。例如，函数$(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4)$在8个点上为真，而且它有正交DNF

$$(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_2 \wedge x_3 \wedge x_4)$$

换句话说，重叠的子立方体11**，*11*，**11可以被替代为不相交的子立方体11**，011*，*011。利用习题30中子立方体的二进制标记法，这些子立方体有星号码0011，0001，1000以及比特码1100，0110，0011。

当星号码分别是 $\overline{B_1},\cdots,\overline{B_p}$ 时，每一个单调函数都可以通过一个比特码列表 $B_1,\cdots,B_p$来定义。给定该列表，令B_k的“投影”S_k为 $B_j \mathbin{\&} \overline{B_k}$ 的按位或，其中对所有的$1 \leqslant j<k$有 $\nu(B_j \mathbin{\&} \overline{B_k})=1$：

$$S_k=\beta_{1k} \mid \cdots \mid \beta_{(k-1)k},\ \beta_{jk}=((B_j \mathbin{\&} \overline{B_k}) \oplus ((B_j \mathbin{\&} \overline{B_k})-1)) \dot{-} ((B_j \mathbin{\&} \overline{B_k})-1)$$

例如，当比特码是(B_1, B_2, B_3)=(1100, 0110, 0011)时，可得投影码(S_1, S_2, S_3)=(0000, 1000, 0100)。

a) 证明星号码 $A'_j=\overline{B_j}-S_j$和比特码B_j定义了与星号码 $A_j=\overline{B_j}$ 对应的子立方体覆盖了相同点的子立方体。

b) 比特码列表$B_1, \cdots, B_p$称为去壳的（shelling），如果对所有$1 \leqslant j<k \leqslant p$，$B_j \& S_k$非零。例如，(1100, 0110, 0011)是一个去壳的；但是如果我们将那些比特码排列为(1100, 0011, 0110)，那么去壳的条件在j=1和k=2时不成立，虽然S_3=1001。证明：(a) 中的子立方体是不相交的当且仅当那个比特码列表是一个去壳的。

c) 根据定理Q，当我们以这种方式阐述一个单调的布尔函数时，每个基本蕴涵项都出现在B中。但是如果想要子立方体是不相交的，有时我们需要添加额外的蕴涵项。例如，比特码1100和0011是没有去壳的。证明我们可以通过添加一个比特码来为函数 $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$获得一个去壳的。得到的正交DNF是什么？

d) 重排比特码{11000, 01100, 00110, 00011, 11010}以得到去壳的列表。

e) 向集合{110000, 011000, 001100, 000110, 000011}里添加两个比特码来得到一个可去壳的（shellable）的列表。

36. [M21]接习题35，令f为任意一个单调函数，且不恒等于1。证明位向量集合

$$B=\{x \mid f(\boldsymbol{x})=1 且 f(\boldsymbol{x}')=0\},\ \boldsymbol{x}'=\boldsymbol{x}\&(\boldsymbol{x}-1)$$

当它是按字母递减的顺序列出时，总是可去壳的。（向量$\boldsymbol{x}'$是通过将$\boldsymbol{x}$最右端的1变为0得到的。）例如，这种方法从列表(1100, 1011, 0111, 0011)中产生一个$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$的正交DNF。

▶ 37. [M31]为 $(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \cdots \vee (x_{2n-1} \wedge x_{2n})$ 找一个含有2^n-1个蕴涵项的可去壳的DNF，并证明该DNF是含有最少蕴涵项的正交DNF。

38. [05]要检验析取范式函数的可满足性是否是困难的？

▶ 39. [25]令$f(x_1, \cdots, x_n)$为一个表现为带有N个内顶点和N+1个叶子的扩展的二叉树形式的布尔方程式。每一个叶子被一个变量x_k标记，而且每一个内顶点被表1中的16个二元运算中的一个标记；从底到顶运用运算符来产生根的值$f(x_1, \cdots, x_n)$。

解释如何在3CNF中构建一个方程式$f(x_1, \cdots, x_n, y_1, \cdots, y_N)$，它恰有$4N+1$个子句，使得$f(x_1, \cdots, x_n)=\exists y_1, \cdots, \exists y_N F(x_1, \cdots, x_n, y_1, \cdots, y_N)$。（此时$f$是可满足的当且仅当$F$是可满足的。）

40. [23]给定一个无向图G，在布尔变量$\{p_{uv} \mid u \neq v\} \cup \{q_{uvw} \mid u \neq v,\ u \neq w,\ v \neq w,\ u \neq w\}$上构建以下几项，其中$u$、$v$和$w$表示$G$的顶点。

$$A=\wedge\{(p_{uv} \vee p_{uv}) \wedge (\overline{p}_{uv} \vee \overline{p}_{vu}) \mid u \neq v\}$$

$$B=\wedge\{(\overline{p}_{uv} \vee \overline{p}_{vw} \vee p_{uw}) \mid u \neq v,\ u \neq w,\ v \neq w\}$$

$C=\wedge\{(\bar{q}_{uvw}\vee q_{uv})\wedge(\bar{q}_{uvw}\vee p_{vw})\wedge(q_{uvw}\vee\bar{p}_{uv}\vee\bar{p}_{vw})|u\neq v,\ u\neq w,\ v\neq w,\ u\not\to w\}$

$D=\wedge\{(\vee_{v\notin\{u,w\}}(q_{uvw}\vee q_{wvu}))|u\neq w,\ u\not\to w\}$

证明方程式$A\wedge B\wedge C\wedge D$是可满足的当且仅当$G$有一条哈密顿回路。提示：将$p_{uv}$看成'$u<v$'。

41. [20]（鸽巢原理）圣瑟里夫岛包含m只鸽子和n个洞。找出一个合取范式使它是可满足的当且仅当每只鸽子是至少一个洞的唯一占有者。

42. [20]找出一个短的，不是完全平凡的不可满足CNF，即使它包含所有同时是Krom子句的Horn子句。

43. [20]是否存在一种有效的方法来判断一个包含所有Horn子句和/或者Krom子句的合取范式的满足性（可能是混合的）？

44. [M23]通过研究（33）中的含义来完成定理H的证明。

45. [M20] (a)证明恰有一半的n元Horn函数是确切的。

(b) 证明n元Horn函数比n元单调函数多（除非$n=0$）。

46. [20] 在上下文无关法(34)中的11×11个字符对xy中，哪些对会彼此相邻？

47. [20]就像在算法2.2.3T（拓扑排序）中，给定一关系序列 $j\prec k$，其中$1\leqslant j,\ k\leqslant n$，考虑子句

$$x_{j1}\wedge\cdots\wedge x_{jk}\Rightarrow x_k,\ \text{对于}1\leqslant k\leqslant n$$

其中$\{j_1,\ \cdots,\ j_t\}$是使得 $j_i\prec k$ 的元素集合。比较算法C和算法2.2.3T在这些子句上的行为。

▸ 48. [21]测试一组Horn子句的可满足性的好方法是什么？

49. [22]证明：如果$f(x_1,\ \cdots,\ x_n)$和$g(x_1,\ \cdots,\ x_n)$都是按Horn子句定义的CNF，那么将有一种简单的方法来测试是否对所有的$x_1,\ \cdots,\ x_n$都有$f(x_1,\ \cdots,\ x_n)\leqslant g(x_1,\ \cdots,\ x_n)$。

50. [HM42] n元的Horn子句可能有$(n+2)2^{n-1}$种。随机选取其中的$c\cdot 2^n$种，允许重复，其中$c>0$；又令$P_n(c)$表示所有选取的子句同时是可满足的的概率。证明

$$\lim_{n\to\infty}P_n(c)=1-(1-e^{-c})(1-e^{-2c})(1-e^{-4c})(1-e^{-8c})\cdots$$

▸ 51. [22]有许多两人游戏可以通过具体指定的一种有向图来定义，其中的每个顶点代表一个游戏位置。有两个游戏玩家Alice和Bob，他们通过从某个顶点出发并且轮流延伸路径的方法建立一条有向路径，一次只能延伸一条弧。游戏开始前，每个顶点要么被标记为A（意味着Alice赢）或者B（意味着Bob赢），或者C（意味着cat赢），或者是未被标记过的。

当路径达到一个标记着A或者B的顶点v时，则那个玩家获胜。如果v已经被访问过了则游戏停止且没有胜者，然后当前同一个玩家继续开始移动。如果v标记为C，那么当前的玩家有接受平局的选择；否则他必须选择一条外向弧来延伸路径，同时另一个玩家开始游戏。（如果v是个未被标记过的顶点且出度为0，那么当前玩家输掉比赛。）

将四个命题变量$A^+(v)$，$A^-(v)$，$B^+(v)$，$B^-(v)$和图中的每个顶点v联系起来，说明一下如何建立一组有限Horn子句使得$A^+(v)$在核中当且仅当路径从v开始且Alice

先移动时她可以强迫取胜；$A^-(v)$在核中当且仅当Bob可以强迫她输掉比赛；$B^+(v)$和$B^-(v)$与$A^+(v)$和$A^-(v)$类似，只是角色换过来了。

52. [25]（布尔游戏）任何布尔函数$f(x_1, \cdots, x_n)$可用下面的方法导向一种叫做“向前两步或向后一步”的游戏：有两个玩家，0和1，他们重复给变量x_j赋值；y玩家试图使$f(x_1, \cdots, x_n)$等于y。刚开始时所有的变量都未被赋值，并且记号为m的位置为0。玩家轮流玩，并且当前的玩家或者令$m \leftarrow m+2$（如果$m+2 \leqslant n$），或者$m \leftarrow m-1$（如果$m-1 \geqslant 1$），然后令

$$\begin{cases} x_m \leftarrow 0\text{或}1, & \text{如果}x_m\text{没有预先设定} \\ x_m \leftarrow \bar{x}_m, & \text{如果}x_m\text{预先设定} \end{cases}$$

当所有变量都被赋值后游戏结束；此时$f(x_1, \cdots, x_n)$为赢家。如果同样的状态（包括m的值）达到两次则为平局。注意任何时刻最多能移动4步。

研究当$2 \leqslant n \leqslant 9$时，以下四种情况下的游戏。

a) $f(x_1, \cdots, x_n)=[x_1 \cdots x_n < x_n \cdots x_1]$ (按字母顺序)；

b) $f(x_1, \cdots, x_n)=x_1 \oplus \cdots \oplus x_n$；

c) $f(x_1, \cdots, x_n)=[x_1 \cdots x_n$不包含两个连续1]；

d) $f(x_1, \cdots, x_n)=[(x_1 \cdots x_n)_2$是素数]。

53. [23]证明：只要改变下列哪一位的要求(a) Tomlin，(b) Unwin，(c) Vegas，(d) Xie，(e) Yankovic，(f) Zany，就可以安排（37）中不可能进行的喜剧节目演出日程。

54. [20]令$S=\{u_1, u_2, \cdots, u_k\}$为有向图某个强连通分量里的一个文字集合，该有向图和图6中的一个2CNF方程式对应。证明：S同时包含一个变量和它的补，当且仅当对于$2 \leqslant j \leqslant k$中的某些$j$有$u_j=\bar{u}_1$。

▶ 55. [30]称$f(x_1, \cdots, x_n)$为一个更名的Horn函数，如果有布尔常量$y_1, \cdots, y_n$使得$f(x_1 \oplus y_1, \cdots, x_n \oplus y_n)$为一个Horn函数。

a) 给定CNF $f(x_1, \cdots, x_n)$，说明如何建立2CNF $g(y_1, \cdots, y_n)$，使得$f(x_1 \oplus y_1, \cdots, x_n \oplus y_n)$的子句为Horn子句当且仅当$g(y_1, \cdots, y_n)=1$。

b) 设计一种算法使得在$O(m)$步内判断一个给定的长度为m的CNF的所有子句是否可以通过对某个变量子集取补转为Horn子句。

▶ 56.[20]布尔函数$f(x_1, x_2, \cdots, x_n)$的可满足性问题可以在形式上叙述成量化公式

$$\exists x_1 \exists x_2 \cdots \exists x_n f(x_1, x_2, \cdots, x_n)$$

是否为真的问题；这里‘$\exists x_j \alpha$’的意思是“存在一个布尔变量x_j使α成立。”

当我们将一个或更多的存在量词$\exists x_j$替换为全称量词$\forall x_j$时，一个更加一般的求值问题出现了，其中‘$\forall x_j \alpha$’的意思是“对所有的布尔变量x_j，α成立。”

当$f(x, y, z)=(x \vee y) \wedge (\bar{x} \vee z) \wedge (y \vee \bar{z})$时，以下8个量化公式中哪些是对的。

$$\exists x \exists y \exists z f(x, y, z), \exists x \exists y \forall z f(x, y, z), \cdots, \forall x \forall y \forall z f(x, y, z)?$$

▶ 57. [30]（B. Aspvall、M. F. Plass和R. E. Tarjan）接习题56，设计一种算法来在线性时间内判断一个给定的完全量化公式$f(x_1, \cdots, x_n)$是否为真，其中f是任何的

2CNF方程式（任何合取Krom子句）。

▸58. 接习题57，设计一种有效算法来判断一个给定的完全量化的合取Horn子句是否为真。

▸59. [M20]（D. Pehoushed和R. Fraer, 1997）如果$f(x_1, x_2, \cdots, x_n)$的真值表在恰好k个地方有一个1，证明完全量化公式$Qx_1Qx_2\cdots Qx_n f(x_1, x_2, \cdots, x_n)$中恰好有$k$个为真，其中每个$Q$或者是或者是∀。

60. [12]以下表达式中哪些像（43）中定义的那样产生中数$\langle xyz\rangle$?

(a) $(x\wedge y)\oplus(y\wedge z)\oplus(x\wedge z)$ (b) $(x\vee y)\oplus(y\vee z)\oplus(x\vee z)$

(c) $(x\oplus y)\wedge(y\oplus z)\wedge(x\oplus z)$ (d) $(x\equiv y)\oplus(y\equiv z)\oplus(x\equiv z)$

(e) $(x\overline{\wedge} y)\wedge(y\overline{\wedge} z)\wedge(x\overline{\wedge} z)$ (f) $(x\overline{\wedge} y)\vee(y\overline{\wedge} z)\vee(x\overline{\wedge} z)$

61. [13]判断正误：如果∘是表1中的任何一个二元布尔运算，则有分配律：$w\circ\langle xyz\rangle=\langle(w\circ x)(w\circ y)(w\circ z)\rangle$。

62. [25]（C. Schensted）如果$f(x_1, \cdots, x_n)$是一个单调布尔函数且$n\geqslant 3$。证明中数扩展公式：

$$f(x_1,\cdots,x_n)=\langle f(x_1, x_1, x_3, x_4,\cdots,x_n)f(x_1, x_2, x_2, x_4,\cdots,x_n)f(x_3, x_2, x_3, x_4,\cdots,x_n)\rangle。$$

63. [20]等式（49）显示了如何通过3个元素的中数来计算5个元素的中数。相反地，我们可以通过5个元素的中数的子例程来计算$\langle xyz\rangle$吗?

64. [23]（S. B. Akers, Jr）(a) 证明一个布尔函数$f(x_1, \cdots, x_n)$是单调且自对偶的，当且仅当它满足以下条件：

对所有的$x=x_1\cdots x_n$和$y=y_1\cdots y_n$存在一个k使得$f(x)=x_k$和$f(y)=y_k$。

(b) 假定f对于一些值未定义，但是只要$f(x)$和$f(y)$都同时定义了则上述条件成立。证明存在一个单调自对偶的布尔函数g满足只要当$f(x)$被定义过了就有$g(x)=f(x)$。

▸65. [M21]任何$\{1, 2, \cdots, n\}$的子集X通过规定$x_j=[j\in X]$对应于一个二进制向量$\boldsymbol{x}=x_1x_2\cdots x_n$。而且任何这样的子集族$\mathcal{F}$通过规定$f(x)=[X\in\mathcal{F}]$对应于一个$n$元布尔函数$f(x)=f(x_1x_2\cdots x_n)$，因此每一个关于子集族的陈述对应于一个关于布尔函数的陈述，反过来也一样。

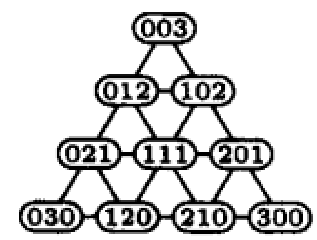
如果一个族$\mathcal{F}$满足只要$X, Y\in\mathcal{F}$则有$X\cap Y\neq\varnothing$，那么F称为相交的（intersecting）。如果一个相交的族只要添加另一个子集后就不相交则称这个族为极大的。证明$\mathcal{F}$是一个极大相交族当且仅当相应的布尔函数f是单调且自对偶的。

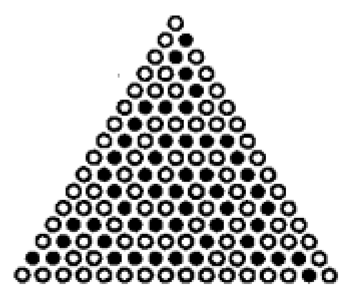
▸66. [M25] $\{1, \cdots, n\}$上的一个仲裁群（coterie）是若干称为仲裁集（quorum）的子集族C，有以下性质：当$Q\in C$和$Q'\in C$：(i) $Q\cap Q'\neq\varnothing$ (ii) $Q\subseteq Q'$蕴涵$Q\neq Q'$。如果$C\neq C'$，称仲裁群C统治（dominate）仲裁群C'，而且对每个$Q'\in C'$，存在一个$Q\in C$使得$Q\subseteq Q'$。比如，仲裁群$\{\{1,2\},\{2,3\}\}$是被$\{\{1,2\},\{1,3\},\{2,3\}\}$统治的，也同时被$\{\{2\}\}$统治。[在L. Lamport的经典论文里介绍了仲裁群，《CACM》21 (1978), 558—565; H. Garcia-Molina和D. Barbara，《JaCM》32 (1985), 841—860。对于分布式系统协议，包括资源互斥、数据复制和名字服务等，它们有很多应用。

在这些应用中C比任何被它统治的仲裁群更易被接受。]

证明C是一个未被统治仲裁群当且仅当它的仲裁集是一个单调自对偶布尔函数$f(x_1, \cdots, x_n)$的基本蕴涵项里的变量的指标集。(这样表2显示了在$\{1, 2, 3, 4\}$上的未被统治仲裁群。)

▶ 67. [M30] (J. W. Milnor和C. Schensted)一个n阶三角网格，这里显示的是n=3的情况，包括$(n+2)(n+1)/2$个带有非负“重心坐标”xyz的点，其中$x+y+z=n$。两个点是相邻的，如果它们恰好有两个坐标位置相差±1。一个点被称为落在x边上，如果它的x坐标为0；被称为落在y边上，如果它的y坐标为0；或者落在z边上，如果它的z坐标为0。因此每边包含$n+1$个点。如果$n>0$，一个点落在两条不同边上当且仅当它是三个顶点之一。





一个“Y”是一个每条边上至少有一个点的点的连通集。假定一个三角网格的每个顶点被一个白的或黑的石子覆盖。比如，52个黑石子中包含一个(有点变形)Y；但是如果它们中的任何一个从黑的变为白的，就有一个白的Y。经过一段时间的思考，我们就会从直观上清楚地看到，在任何布置中，黑石子包含一个Y当且仅当白石子不包含。

我们可以将每个石子的颜色用一个布尔变量来代替，0表示白色而1表示黑色。令$Y(t)=1$当且仅当有一个黑Y，其中t是一个由所有的布尔变量构成的三角网格。这个函数Y显然是单调的；而且上一段中的直观声明等价于说Y也是自对偶的。这一题的目的就是为了利用中数代数严格证明那个声明。

给定$a, b, c \geqslant 0$，令t_{abc}为包含所有坐标xyz满足$x \geqslant a$, $y \geqslant b$, $z \geqslant c$的点的三角子网格，t_{001}就是除了在z边上(最下面一排)的所有点。注意，如果$a+b+c=n$，t_{abc}就是坐标为abc的那个点；一般地，t_{abc}是一个$n-a-b-c$阶三角网格。

a) 证明：如果$n>0$，$Y(t)=\langle Y(t_{100})Y(t_{010})Y(t_{001})\rangle$。

b) 如果$n>0$，令t^*为$n-1$阶三角网格，它由以下规则定义

$$t^*_{xyz}=\langle t_{(x+1)yz}t_{x(y+1)z}t_{xy(z+1)}\rangle,\ \text{对于}x+y+z=n-1。$$

证明$Y(t)=Y(t^*)$。[换句话说，t^*通过取它们颜色的中间值来压缩每个石子的小三角。重复这个过程定义了一个石子金字塔，它的顶点为黑石子当且仅当在底部有一个黑Y。将这个压缩原理应用到上面的变形的Y上是很有趣的。]

68. [46]习题67中显示的刚好勉强构成Y的布置有52个黑石子。这样的布置里最多能有多少个黑石子？(也就是函数$Y(t)$的一个基本蕴涵项里能有几个变量？)

▶ 69. [M26] (C. Schensted)习题67从中数的角度来描述Y函数。相反地，令

$f(x_1,\cdots,x_n)$为任何一个具有$m+1$个基本蕴涵项p_0, p_1, …, p_m的单调自对偶布尔函数。证明$f(x_1,\cdots,x_n)=Y(T)$，其中T是任何一个$m-1$阶三角网格，只要它满足其中的T_{abc}是p_a和p_{a+b+1}的共有变量，其中a、b、c满足$a+b+c=m-1$。比如，当$f(w,x,y,z)=\langle xwywz\rangle$时我们有$m=3$且

$$f(w,x,y,z)=(w\wedge x)\vee(w\wedge y)\vee(w\wedge z)\vee(x\wedge y\wedge z)=Y\begin{pmatrix} & w & \\ w & & w \\ x & y & z\end{pmatrix}。$$

▶ 70. [M20]（A. Meyerowitz，1989）给定任一个单调自对偶布尔函数$f(x)=f(x_1,\cdots,x_n)$，选择任意一个基本蕴涵项$x_{j_1}\wedge\cdots\wedge x_{j_s}$并且令

$$g(x)=(f(x)\wedge[x\neq t]\vee[x=\bar{t}])$$

其中$t=t_1\cdots t_n$是在$\{j_1,\cdots,j_s\}$中有1的位向量。证明$g(x)$也是单调且自对偶的。（注意，除了在t和$\bar{t}$两点外，$g(x)$等于$f(x)$。）

▶ 71. [M21]给定一个中数代数的公理(50)、(51)和(52)，证明长式分配律(54)是短式分配律(53)的一个推论。

72. [M22]请从中数定律(50)～(53)中推导出(58)、(59)和(60)。

73. [M32]（S. P. Avann）给定一个中数代数M，它的区间由(57)定义，并且相应的中数图由(61)定义，令$d(u,v)$表示u到v的距离，同时用$[uxv]$表示“x位于由u到v的一条最短路径上”。

a) 证明$[uxv]$成立当且仅当$d(u,v)=d(u,x)+d(x,v)$。

b) 假设$x\in[u\,..\,v]$且$u\in[x\,..\,y]$，其中$x\neq u$且y—v是图的一条边。证明x—u也是一条边。

c) 如果$x\in[u\,..\,v]$，通过对$d(u,v)$归纳证明$[uxv]$。

d) 相反地，请证明$[uxv]$蕴涵$x\in[u\,..\,v]$。

74. [M21]在一个中数代数中，根据定义(57)证明无论何时只要有$w\in[x\,..\,y]$，$w\in[x\,..\,z]$且$w\in[y\,..\,z]$，则$w=\langle xyz\rangle$。

▶ 75. [M36]（M. Sholander，1954）假设M是一个点集，其上定义有介于关系（betweenness relation）“x位于u和v之间”，用$[uxv]$表示，该关系满足以下三个公理：

i) 如果$[uvu]$，则$u=v$。

ii) 如果$[uxv]$且$[xyu]$，则$[vyu]$。

iii) 给定x、y和z，恰有一点$w=\langle xyz\rangle$满足$[xwy]$、$[xwz]$和$[ywz]$。

这个习题的目的是证明M是一个中数代数。

a) 证明众数律$\langle xxy\rangle=x$，也就是(50)。

b) 证明交换律$\langle xyz\rangle=\langle xzy\rangle=\cdots=\langle zyx\rangle$，也就是(51)。

c) 证明$[uxv]$当且仅当$x=\langle uxv\rangle$。

d) 如果有$[uxy]$且$[uyv]$，证明$[xyv]$。

e) 如果有$[uxv]$、$[uyz]$且$[vyz]$，证明$[xyz]$。提示：构造点$w=\langle yuv\rangle$，$p=\langle wux\rangle$，$q=\langle wvx\rangle$，$r=\langle pxz\rangle$，$s=\langle qxz\rangle$和$t=\langle rsz\rangle$。

f) 最后，推导短式分配律，也就是(53)：$\langle\langle xyz\rangle uv\rangle=\langle x\langle yuv\rangle\langle zuv\rangle\rangle$。

76. [M33]推导习题75中关于介于关系的公理(i)、(ii)、(iii)，请从3个中数公理(50)、(51)和 (52)开始，并令$[uxv]$为"$x=\langle uxv\rangle$"的缩写。不使用分配律(53)。提示：参见习题74。

77. [M28]令G为一个包含边r—s的中数图。对每条边u—v，我们把u叫做v的一个早期邻居（early neighbor）当且仅当u比v更接近r。将所有的顶点分割成"左""右"两个部分，左边的部分相对于s更接近于r，右边的部分相对于r更接近于s。每个右边的顶点v都有一个秩（rank），它是v 到一个左顶点的最短距离。类似地，每个左顶点u都有一个秩1—d，d是从u到右顶点的最短距离。因此u的秩为零如果它邻接于一个右顶点，否则它的秩就是负值。顶点r 显然有秩0，而 s有秩1。

a) 请证明每一个秩为1的顶点恰好邻接一个秩为0的顶点。

b) 证明所有右侧顶点的集合是凸的。

c) 证明所有秩为1的顶点集合是凸的。

d) 证明子程序I的步骤I3—I9正确地标识出了所有秩为1和秩为2的顶点。

e) 证明算法H是正确的。

▶ 78. [M26]如果在执行算法H时，顶点v在步骤I4上被检测了k次，证明这个图最少有2^k个顶点。提示：有k个方法来开始一条从v到a 的最短路径；因此最少有k个l出现在$l(v)$中。

▶ 79. [M27]（R. L. Graham）一个超正方体的子图是一个其顶点v可以按一定方法用位串$l(v)$标记的图。该方法是u—v当且仅当$l(u)$和$l(v)$恰好在一个比特位置上不同。(每个标记都有相同的长度。)

a) 一种定义一个超正方体的 n-顶点子图的方法是令$l(v)$为v的二进制表示，其中$0\leqslant v<n$。证明这个子图正好有 $f(n)=\sum_{k=0}^{n-1}\nu(k)$条边，其中$\nu(k)$是横向加法（sideways addition）函数。

b) 证明$f(n)\leqslant n\lceil\lg n\rceil/2$。

c) 证明超正方体中没有一个n-顶点子图会多于$f(n)$条边。

80. [27]一个部分立方体（partial cube）是一个超正方体的"等比例的"子图，也就是这个子图中顶点之间的距离和在全图中是相同的。这个部分立方体的顶点因此可以按如下方法被标记：从u到v的距离就是$l(u)$和$l(v)$间的"海明距离"，也就是$\nu(l(u)\oplus l(v))$。

算法H表明了每个中数图都是一个部分立方体。

a) 找到一个不是部分立方体的4-立方体子图。

b) 给出一个不是中数图的部分立方体的例子。

81. [16] 每个中数图都是二分图吗？

82. [25]（服务中的增量变化，incremental changes in service）给定图G中的一个顶点序列$(v_0, v_1, \cdots, v_t)$，考虑问题：找到另外一个序列$(u_0, u_1, \cdots, u_t)$使$u_0=v_0$，且总和$(d(u_0, u_1)+d(u_1, u_2)+\cdots+d(u_{t-1}, u_t)+(d(u_1, v_1)+d(u_2, v_2)+\cdots+d(u_t, v_t))$最小，其中$d(u, v)$表示从$u$到$v$的距离。（每个$v_k$都可以看作是一个在该顶点对需要资源的请

求；一个服务器如那些请求被按顺序处理一样移动到u_k。）证明如果G是一个中数图，则对于$0<k<t$，$u_t=v_t$，我们可以通过选取$u_k=\langle u_{k-1}v_kv_{k+1}\rangle$得到一个最优解。

▶ 83. [28] 对习题82进行推广，在中数图中，找出一个对$(d(u_0, u_1)+d(u_1, u_2)+\cdots+d(u_{t-1}, u_t)+\rho(d(u_1, v_1)+d(u_2, v_2)+\cdots+d(u_t, v_t))$进行最小化的有效方法，其中$\rho$为任一给定的正比率。

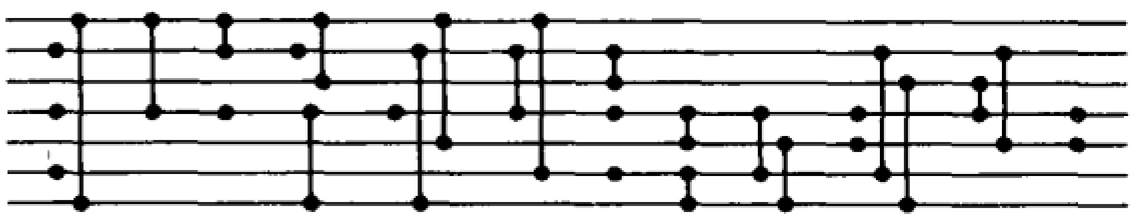
84. [30]编写一个程序，找出所有5元单调自对偶布尔函数。请问相应的中数图的边是什么？（表2列出4变量的情况）。

▶ 85. [M22]定理S告诉我们每个2CNF公式都对应于一个中数集合；因此如图6中的每个反对称有向图同样也对应于一个中数集合。请指出哪个有向图对应于精简（reduced）中数集？

86. [15]如果v、w、x、y和z属于一个中数集合X，这5个元素的中数$\langle vwxyz\rangle$若按分量计算，是否也总是属于X？

87. [24]定理F的证明中为自由树(63)构造的CI网是什么？

88. [M21]借助并行计算技术，使每个模块在最早的可能时间运行，把网络(74)浓缩成下图



证明，虽然在定理F的证明中，构造的网络可能包含$\Omega(t^2)$个模块，但其最大延迟不会超过$O(t\log t)$级别。

89. [24] 对于某些文字u和v，若(73)的结构中添加了一个新的模块组来执行条件$u\to v$，证明其保存了所有先前生效的条件$u'\to v'$。

▶ 90. [21] 构造一个输入位为$x_1\cdots x_t$且输出位为$y_1\cdots y_t$的CI网，其中$y_1=\cdots=y_{t-1}=0$且$y_t=x_1\oplus\cdots\oplus x_t$。试求它只有$O(\log t)$级别的延时。

91. [46]每个t维中数图的标记的一个收缩映射可以由一个仅有$O(\log t)$级别延时的CI网计算吗？[这个问题是由排序等类似问题的渐进最优网络的存在性引出的。参见M. Ajtai、J. Komlós 和E. Szemerédi，《Combinatorica》3(1983), 1—19]

92. [46] 在对n个布尔输入进行排序时，CI网可以比一个没有反转器的“纯”排序网络使用更少的模块吗？

93. [M20]证明一个图G的每个收缩X都是G的一个等距子图（isometric subgraph）。（换句话说，在X中的距离和在G中的距离一样；参见习题80。）

94. [M21] 如果不考虑那些所有$x\in X$是常量的坐标，证明一个超正方体的每个收缩X都是一个中数标记集合。

95. [M25] 判断正误：当输入为所有可能位串时，一个由比较-反转器网络产生的输出集合总是一个中数集。

96. [HM25]我们可以允许(75)中的常量$w_1, w_2, \cdots, w_n$和t是任意实数而不是要求它们必须是整数，那么那样是否会增加阈函数的数目？

97. [10]当$n=2$，$w_1=w_2=1$，且$t=-1$，0，1，2，3或4时，会出现在(81)中的中数/众数函数是什么？

98. [M23]证明任何自对偶阈函数可以用以下的形式表示

$$f(x_1, x_2, \cdots, x_n)=[v_1y_1+\cdots+v_ny_n>0]$$

其中每个y_j或者是x_j或者是$\bar{x}_j$。比如，$2x_1+3x_2+5x_3+7x_4+11x_5+13x_6 \geqslant 21$当且仅当$2x_1+3x_2+5x_3-7\bar{x}_4+11x_5-13\bar{x}_6>0$。

▸ 99. [20]（J. E. Mezei, 1961）证明

$$\langle\langle x_1 \cdots x_{2s-1}\rangle y_1 \cdots y_{2t-2}\rangle = \langle x_1 \cdots x_{2s-1} y_1^s \cdots y_{2t-2}^s\rangle$$

100. [20]判断正误：如果$f(x_1, \cdots, x_n)$是一个阈函数，那么函数$f(x_1, \cdots, x_n)\wedge x_{n+1}$和$f(x_1, \cdots, x_n)\vee x_{n+1}$也是。

101. [M23]当$n \geqslant 3$时，用公式$\langle x_1^{F_1} x_2^{F_2} \cdots x_{n-1}^{F_{n-1}} x_2^{F_{n-2}}\rangle$定义斐波那契阈函数（Fibonacci threshold function）$F_n(x_1, \cdots, x_n)$，如$F_7(x_1, \cdots x_7)=\langle x_1x_2x_3^2x_4^3x_5^5x_6^8x_7^5\rangle$。

a) $F_n(x_1, \cdots, x_n)$的基本蕴涵项是什么？

b) 为$F_n(x_1, \cdots, x_n)$找到一个正交DNF（参见习题35）。

c) 用Y函数的形式表示$F_n(x_1, \cdots, x_n)$（参见习题67和69）。

102. [M21]一个布尔函数的自对偶性（self-dualization）是由以下公式定义的。

$$\hat{f}(x_0, x_1, \cdots x_n) = (x_0 \wedge f(x_1, \cdots, x_n)) \vee (\bar{x}_0 \wedge \overline{f(\bar{x}_1, \cdots, \bar{x}_n)})$$
$$= (\bar{x}_0 \wedge f(x_1, \cdots, x_n)) \wedge (x_0 \vee \overline{f(\bar{x}_1, \cdots, \bar{x}_n)})$$

a) 如果$f(x_1, \cdots, x_n)$是任意一个布尔函数，证明$\hat{f}$是自对偶的。

b) 证明如果$\hat{f}$是阈函数当且仅当f是阈函数。

103. [HM25]给出若干基本蕴涵项，解释怎样用线性规范方法去测试一个单调自对偶的布尔函数是阈函数。如果是阈函数，怎样最小化其众数函数$\langle x_1^{w_1} \cdots x_n^{w_n}\rangle$的表示形式。

104. [25]用习题103 的方法找出下列阈函数的最简单的众数函数表示形式。

(a) $\langle x_1^2x_2^3x_3^5x_4^7x_5^{11}x_6^{13}x_7^{17}x_8^{19}\rangle$；(b) 对$0 \leqslant t \leqslant 16$（17种情况），$[(x_1x_2x_3x_4)_2 \geqslant t]$；

(c) $\langle x_1^{29}x_2^{25}x_3^{19}x_4^{15}x_5^{12}x_6^8x_7^8x_8^3x_9^3x_{10}\rangle$。

105. [M25]证明在习题101中，定义题中斐波那契阈函数的那个函数是其最简单的众数函数表示形式。

▸ 106. [M25]证明中数操作$\langle x\bar{y}\bar{z}\rangle$为真当且仅当$x \geqslant y+z$。

a) 推广之，试证明通过执行$2^{n+1}-1$个布尔变量的中数，可以测试$(x_1x_2\cdots x_n)_2 \geqslant (y_1y_2\cdots y_n)_2+z$成立的条件。

b) 证明若少于$2^{n+1}-1$个，则没有中数能够满足问题的条件。

107. [17]计算表1中16个函数的$N(f)$和$\Sigma(f)$。（参见定理T。）

108. [M21]设$g(x_0, x_1, \cdots, x_n)$是自对偶函数；因而$N(g)=2^n$是定理T的标记。用$\Sigma(g)$表示$N(f)$和$\Sigma(f)$，当$f(x_1, \cdots, x_n)$满足：

a) $g(0, x_1, \cdots, x_n)$；b) $g(1, x_1, \cdots, x_n)$。

109. [M25]称二进制串$\alpha=a_1a_2\cdots a_n$将$\beta=b_1b_2\cdots b_n$众数化了（majorize），记为$\alpha\succeq\beta$或$\beta\preceq\alpha$，如果对任意$k\geqslant0$有$a_1+\cdots+a_k\geqslant b_1+b_2+\cdots+b_k$。

a) 设$\bar{a}=\bar{a}_1\cdots\bar{a}_n$，证明$\alpha\succeq\beta$当且仅当$\bar{\beta}\succeq\bar{\alpha}$。

b) 证明任意两个长度为n的二进制串有一个最大的下界$\alpha\wedge\beta$，该下界有如下性质：$\alpha\succeq\gamma$且$\beta\succeq\gamma$当且仅当$\alpha\wedge\beta\succeq\gamma$。若给出$\alpha$和$\beta$，怎么计算$\alpha\wedge\beta$。

c) 同样地，如何计算最小的上界$\alpha\vee\beta$，该上界有性质：$\gamma\succeq\alpha$且$\gamma\succeq\beta$当且仅当$\gamma\succeq\alpha\vee\beta$。

d) 判断正误：$\alpha\wedge(\beta\vee\gamma)=(\alpha\wedge\beta)\vee(\alpha\wedge\gamma)$; $\alpha\vee(\beta\wedge\gamma)=(\alpha\vee\beta)\wedge(\alpha\vee\gamma)$。

e) 称α覆盖（cover）β，如果$\alpha\succeq\beta$且$\alpha\neq\beta$，而且如果$\alpha\succeq\gamma\succeq\beta$，则蕴涵$\gamma=\alpha$或$\gamma=\beta$。比如，图8说明了长度为5的二进制串的覆盖关系。给出一个简单的寻找被给定二进制串覆盖的字符串的方法。

f) 证明，若α_{j-1}覆盖α_j（$1\leqslant j\leqslant\gamma$），则从一个给定串$\alpha$到$0\cdots0$生成的每条路径$\alpha=\alpha_0,\alpha_1,\cdots\alpha_r=0\cdots0$有相同长度$r=r(\alpha)$。

g) 设$m(\alpha)$为二进制串β的数目，其中$\beta\succeq\alpha$。证明$m(1\alpha)=m(\alpha)$且$m(0\alpha)=m(\alpha)+m(\alpha')$，其中$\alpha'$是由$\alpha$最左边的1（如果有的话）变为0生成的。

h) 有多少长度为n的串α满足$\bar{\alpha}\succeq\alpha$？

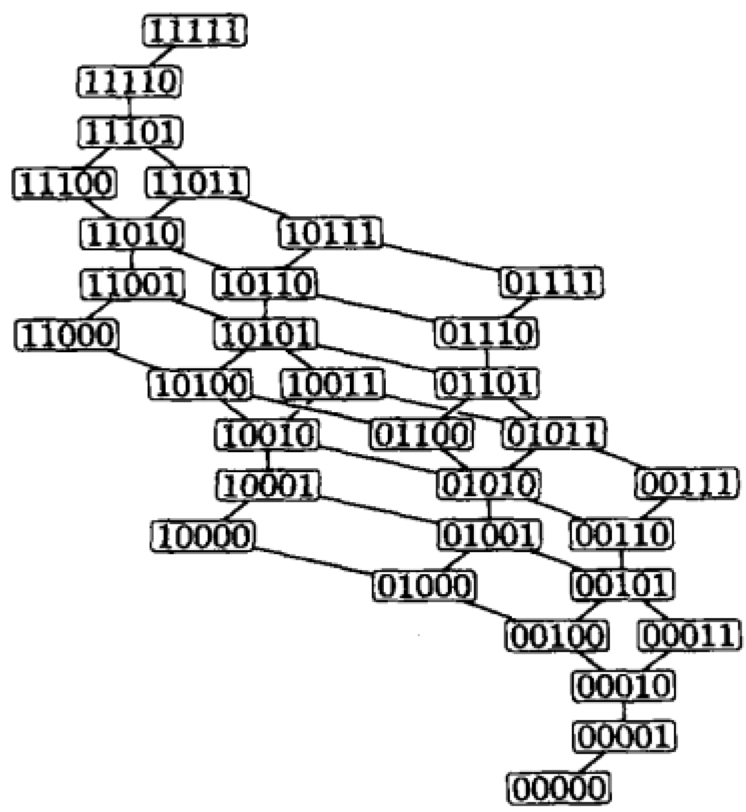


图8 长度为5的串的二进制众数化格（参见习题109）

110. [M23]一个布尔函数称为正则的（regular），对任意的向量x和y，若$x\preceq y$隐含$f(x)\leqslant f(y)$，此处的$\preceq$为习题109中的众数化关系，请证明或推翻以下结论：

a) 每个正则函数是单调的。

b) 如果对于$w_1\geqslant w_2\geqslant\cdots\geqslant w_n$，$f$是（75）中的阈函数，则$f$是正则的。

c) 如果f是（b）中描述的函数，且$\Sigma(f)=(s_1,\cdots,s_n)$，则$s_1\geqslant s_2\geqslant\cdots\geqslant s_n$。

d) 如果f是（75）中的阈函数且$s_1\geqslant s_2\geqslant\cdots\geqslant s_n$，则$w_1\geqslant w_2\geqslant\cdots\geqslant w_n$。

111. [M36]一个工作概率为$(p_1,\cdots,p_n)$的系统最优仲裁群（optimum coterie）是一个这样的仲裁群，它对应于在所有n元单调自对偶函数中具有最大可能性的单调自对偶函数。（参见习题14和66。）

a) 证明：如果$1\geqslant p_1\geqslant\cdots\geqslant p_n\geqslant\dfrac{1}{2}$，那么至少存在一个具有最大可能性的自对偶函数是常函数，并描述这一函数。

b) 进一步证明，它可以测试一个自对偶常函数f在二元优化格的点y上的最优性，其中y满足：对所有的被y覆盖的x有$f(y)=1$但$f(x)=0$。

c) 当有一些概率$<\dfrac{1}{2}$时，什么仲裁群是最优的？

▸ 112. [M37]（J. Håstad）如果$f(x_1,x_2,\cdots,x_m)$是布尔函数，令$M(f)$是它的系数

多重线性表示（参见习题12），利用Chase序列$\alpha_0=00\cdots0$, $\alpha_1=10\cdots0$, $\cdots$, $\alpha_{2^m-1}=11\cdots1$，重新排列该多项式的项，使它按指数排列；Chase序列是通过串联7.2.1.3节的(35)中的序列A_{n0}, $A_{(n-1)1}$, $\cdots$, A_{0n}生成的，它有如下好的性质：一个小小的改变就可以使α_j和α_{j+1}完全一样，或者是0→1或01→10或001→100或10→01或100→001。比如，有一个Chase序列：

0000，1000，0010，0001，0100，1100，1010，1001，0011，0101，0110，1110，1101，1011，0111，1111

当m=4时，对应各自的项是1, x_1, x_3, x_4, x_2, x_1x_2, $\cdots$, $x_2x_3x_4$, $x_1x_2x_3x_4$；所以，当项以这种方式排列时，$((x_1\oplus\bar{x}_2)\wedge x_3)\vee((x_1\wedge\bar{x}_3)\wedge x_4)$的多重线性表示为$x_3-x_1x_3+x_1x_4-x_2x_3+2x_1x_2x_3-x_1x_3x_4$。现在，令$F(f)=[M(f)$中最重要的系数是正的]。

比如，按照Chase排序，项$((x_1\oplus\bar{x}_2)\wedge x_3)\vee((x_1\wedge\bar{x}_3)\wedge x_4)$最重要的（最终的）非零项为$-x_1x_3x_4$，所以在此例中$F(f)=0$。

a) 求表1中的16个函数的$F(f)$值。

b) 证明当f是其真值表的$n=2^m$项$\{f_{0\cdots00}, f_{0\cdots01}, \cdots, f_{1\cdots11}\}$时，函数$F(f)$是阈函数。当$m$=2时，写出该函数。

c) 证明，当m很大时，任何阈函数F的权重也一定很大，它们的绝对值应该超过

$$\frac{3^{\binom{m}{3}}7^{\binom{m}{4}}15^{\binom{m}{5}}\cdots(2^{m-1}-1)^{\binom{m}{m}}}{n}(1-O(n^{-1}))=2^{mn/2-n-2(3/2)^m/\ln 2+O((5/4)^m)}$$

提示：考虑真值表项的离散傅里叶变换。

113. [24]证明使用下列三种阈操作就足以计算（91）中$S_{2,3,6,8,9}(x_1, \cdots, x_{12})$的值：

$g_1(x_1,\cdots,x_{12})=(\nu x\geqslant 6)=\langle 1x_1\cdots x_{12}\rangle$;

$g_2(x_1,\cdots,x_{12})=(\nu x-6g_1\geqslant 2)=\langle 1^3x_1\cdots x_{12}\bar{g}_1^6\rangle$;

$g_3(x_1,\cdots,x_{12})=(-2\nu x+13g_1+7g_2\geqslant 1)=\langle 0^5\bar{x}_1^2\cdots\bar{x}_{12}^2g_1^{13}g_2^7\rangle$。

同时找出一个计算$S_{1,3,5,8}(x_1, \cdots, x_{12})$的4种阈操作模式。

114. （D. A. Huffman）函数$S_{3,6}(x, x, x, x, y, y, z)$是什么？

115. [M22]解释为何习题（92）能够正确计算奇偶函数$x_0\oplus x_1\oplus\cdots\oplus x_{2m}$。

▸116. [HM28]（B. Dunham和R. Fridshal, 1957）考虑对称函数，可以证明n元的布尔函数可能有很多的基本蕴涵项。

a) 假设$0\leqslant j\leqslant k\leqslant n$，问哪些对称函数$f(x_1, \cdots, x_n)$包含基本蕴涵项$x_1\wedge\cdots\wedge x_j\wedge\bar{x}_{j+1}\wedge\cdots\wedge\bar{x}_k$？

b) 函数$S_{3,4,5,6}(x_1,\cdots,x_9)$有多少个基本蕴涵项？

c) 令$\hat{b}(n)$为所有n元对称布尔函数的基本蕴涵项的最大数目，找到一个计算$\hat{b}(n)$的递推公式，并计算$\hat{b}(9)$。

d) 证明$\hat{b}(n)=\Theta(3^n/n)$。

e) 进一步证明，存在对称函数$f(x_1,\cdots,x_n)$，它对f和$\bar{f}$均有$\Theta(2^{3n/2}/n)$个基本蕴涵项。

117. [M26]一个析取范式称为是非冗余的，如果它没有一个蕴涵项蕴涵其他的

蕴涵项。在所有n元布尔函数中，令$b^*(n)$是一个非冗余DNF的最大蕴涵项数目，找出一个$b^*(n)$的简单表达式，并求出它的渐近值。

118. [29]对$m=0, 1, \cdots$有多少个布尔函数$f(x_1, x_2, x_3, x_4)$正好有m个基本蕴涵项？

119. [M48]接习题117，令$b(n)$是一个n元布尔函数的基本蕴涵项的最大数目，显然$\hat{b}(n)\leqslant b(n)\leqslant b^*(n)$，求$b(n)$的渐近值。

120. [23]求下列对称函数的最短DNF。

(a) $x_1\oplus x_2\oplus\cdots\oplus x_n$；

(b) $S_{0,1,3,4,6,7}(x_1,\cdots,x_7)$；

(c) 证明任意一个n元的布尔函数可以表示成至多包含2^{n-1}个基本蕴涵项的DNF。

▸121. [M23]函数$\langle 1(x_1\oplus x_2)y_1y_2y_3\rangle$是部分对称的，因为它在$\{x_1,x_2\}$和$\{y_1,y_2,y_3\}$上是对称的，而在所有5个变量$\{x_1,x_2,y_1,y_2,y_3\}$上是不对称的，求解：

a) 有多少个布尔函数$f(x_1,\cdots,x_m,y_1,\cdots,y_n)$在$\{x_1,x_2,\cdots,x_m\}$和$\{y_1,y_2,\cdots,y_n\}$上是对称的？

b) 有多少是单调的？

c) 有多少是自对偶的？

d) 有多少是既单调又自对偶的？

122. [M25]接习题110和121，找到所有布尔函数$f(x_1,x_2,x_3,y_1,y_2,y_3,y_4,y_5,y_6)$，它在$\{x_1,x_2,x_3\}$和$\{y_1,y_2,\cdots,y_6\}$上既是对称又是对偶的正则函数。其中有哪些是阈函数？

123. [46]确定10元阈函数的自对偶布尔函数的数目。

124. [20]在表5的基本规则下，找出一个等于767个其他函数的4元布尔函数。

125. [18]（95）中的函数集中哪个是渠限化的？

126. [23]（a）证明一个布尔函数是渠限化的当且仅当其基本蕴涵项和基本子句上有某个简单的性质。（b）证明一个布尔函数是渠限化的当且仅当它们的Chow参数$N(f)$和$\Sigma(f)$有某个简单的性质（参见定理T）。（c）类比于整数向量$\Sigma(f)$，定义布尔向量$\vee(f)=\bigvee\{x\mid f(x)=1\}$及$\wedge(f)=\bigwedge\{x\mid f(x)=1\}$。证明只要给出向量$\vee(f)$，$\vee(\bar{f})$，$\wedge(f)$和$\wedge(\bar{f})$就能推出$f$是不是渠限化的。

127. [M25]哪些渠限化函数是（a）自对偶的？（b）是确切Horn函数？

▸128. [20]找出一个非渠限化函数$f(x_1,\cdots,x_n)$，它刚好在两点上为真。

129. [M25]n元函数中有多少个是渠限化函数？

130. [M21]根据表3，总共有168个4元单调布尔函数，但是有一些，像$x\wedge y$，只依赖于3个或更少的变量，问

a) 涉及每个变量的4元单调布尔函数有几个？

b) 如表4，若不关心排列，有多少个函数是唯一的？

131. [HM42]表3清楚表明Horn函数的数目要比Krom函数的数目多得多。问当$n\to\infty$时，渐近值是多少？

▸132. [HM30]布尔函数$g(x)=g(x_1,\cdots,x_n)$被称为是仿射的（affine），若对某些布尔

常量$y_0,y_1\cdots,y_n$它可以写成形式$y_0\oplus(x_1\wedge y_1)\oplus\cdots\oplus(x_n\wedge y_n)=(y_0+x\cdot y) \bmod 2$。

a) 给定任意布尔函数$f(x)$，证明有一些仿射函数在$2^{n-1}+2^{n/2-1}$个或更多的点上和$f(x)$是吻合的，提示：令$s(y)=\sum_x(-1)^{f(x)+x\cdot y}$，证明对所有向量$z$有$\sum_y s(y)s(y\oplus z)=2^{2n}[z=0\cdots0]$。

b) 布尔函数$f(x)$被称为弯曲的（bent）如果在超过$2^{n-1}+2^{n/2-1}$个点上没有仿射函数与其吻合。证明$(x_1\wedge x_2)\oplus(x_3\wedge x_4)\oplus\cdots\oplus(x_{n-1}\wedge x_n)\oplus h(x_2,x_4,\cdots,x_n)$是弯曲函数，其中$n$是奇数，$h(y_1,y_2,\cdots,y_{n/2})$是任意的。

c) 证明$f(x)$是弯曲函数当且仅当对所有$y\neq0\cdots0$，有$\sum\limits_x(f(x)f(x\oplus y))=2^{n-1}$。

d) 如果一个弯曲函数$y(x_1,x_2,\cdots,x_n)$用如（19）中的多重线性多项式对2取模表示，证明当$r>n/2>1$时，它不会包含项$x_1\cdots x_r$。

▶133. [20]（Mark A. Smith, 1990）假设独立抛投n个硬币可得到n个随机位，第k个硬币得到1的概率是p_k。给出选择$(p_1,p_2,\cdots,p_n)$的一个方法，使$f(x_1,\cdots,x_n)=1$的概率为$(t_0t_1\cdots t_{2^n-1})_2/(2^{2^n}-1)$，其中$t_0t_1\cdots t_{2^n-1}$是布尔函数$f$的真值表（$n$个随机硬币恰好能产生$2^n$位精度的概率）。

总的来说，设计经济实惠的逻辑电路时，
在所有工程方面的考量中，如何使开关部件的数量最少是最重要的。
——H. A. Curtis,
《A New Approach to the Design of Switching Circuits》(1962)

他一定是个伟大的计算者，成功的关键的确就是简单，简单，再简单。
——Henry D. Thoreau，《Walden; or, Life in the Woods》(1854)

7.1.2 布尔求值

我们的下一个目标是研究布尔函数的高效求值方法，和4.6.4节所讨论的多项式求值方法十分相似。研究该主题的一种自然方法就是考虑由基本操作形成的链，这和前面小节所讨论的多项式链非常类似。

一个有n个变量$(x_1, \cdots, x_n)$的函数的一条布尔链（boolean chain）是一个具有如下性质的序列$(x_{n+1}, \cdots, x_{n+r})$，序列中的每一步都是前两步结果的组合。

$$x_i = x_{j(i)} \circ_i x_{k(i)}, \quad 对于n+1 \leqslant i \leqslant n+r \qquad (1)$$

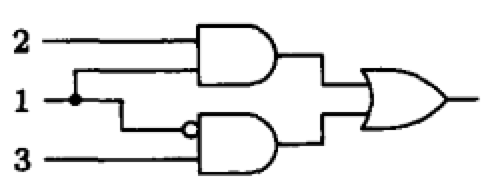
这里$1 \leqslant j(i) < i$，$1 \leqslant k(i) < i$，$\circ_i$是7.1.1节表1中16种二元操作符之一。例如，当$n = 3$时两条链

$$\begin{array}{lcl} x_4 = x_1 \wedge x_2 & & x_4 = x_2 \oplus x_3 \\ x_5 = \bar{x}_1 \wedge x_3 & 和 & x_5 = x_1 \wedge x_4 \\ x_6 = x_4 \vee x_5 & & x_6 = x_3 \oplus x_5 \end{array} \qquad (2)$$

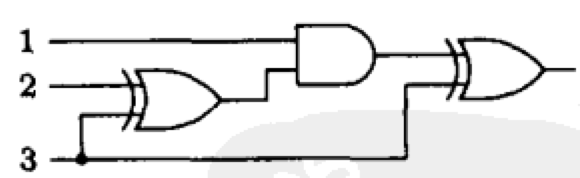
都对“mux”或“if-then-else”函数$x_6 = (x_1\ ?\ x_2 : x_3)$进行求值，根据$x_1$是1(true)或0(false)结果取值$x_2$或$x_3$。

（注意（2）中左边的例子使用简化记法‘$x_5 = \bar{x}_1 \wedge x3$’以表示NOT-BUT操作，而非7.1.1节表1中出现的‘$x_5 = x_1 \subset x_3$’形式。重要的是，不管使用何种记法，布尔链中的每一步都是前两步结果的布尔组合。）

布尔链与电路有着自然的对应关系，链中的每一步都对应一个有两个输入和一个输出的“门”。电气工程师习惯上使用如下的电路图以表示（2）中的布尔链。



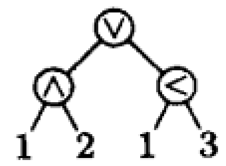
和



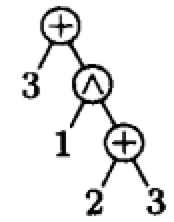
(3)

电气工程师往往需要在各种技术限制下设计经济实惠的电路；如某些门也许比另一些门更为昂贵，一些输出在重用时要进行信号放大，布局可能要求是平面的，某些路径可能要求尽可能短等。但本书主要关心软件而非硬件，所以我们不必担心此类事情。对我们的目标来说，所有的门具有同样的价值，而所有的输出想重用几次就重用几次（用行话来讲，我们的布尔链可归结为所有的门均为有两个扇入并有无限扇出的电路）。

此外，我们将使用如下的二叉树来描述布尔链。



和



(4)

而不是采用如（3）般的电路图。当链的中间若干步被使用超过一次时，对应的二叉树将有重叠的子树。树中每个内部结点使用二元操作符进行标记，而外部结点则使用整数k表示变量x_k，进行标记。（4）左边树中的标记‘⊘’代表NOT-BUT操作符，原因是$\bar{x} \wedge y = [x < y]$；类似地，BUT-NOT运算符$\bar{x} \wedge y$可使用被标记为‘⊘’的结点表示。

一些不同的布尔链可能有着相同的树。例如，（4）中左边的树也可以表示链

$$x_4 = \bar{x}_1 \wedge x_3, \quad x_5 = x_1 \wedge x_2, \quad x_6 = x_5 \vee x_4$$

树结点的任何一个拓扑序都产生一个等价链。

给定一个n变量的布尔函数f，我们经常需要找出一条布尔链$x_{n+r} = f(x_1, \cdots, x_n)$，其中$r$尽可能小。函数$f$的组合复杂度（combinational complexity）$C(f)$是计算它的最短链的长度。为避免过于冗长，简称$C(f)$为“f的代价”。上面例子中的mux函数具有代价3，我们可以通过穷举以说明其不能由任何长度为2的布尔链生成。

7.1.1节研究的f的DNF和CNF表示很少告诉我们关于$C(f)$的事情，本质上更有效的计算方案通常可能存在。例如，在7.1.1节(30)随后的讨论中，我们发现真值表为1100 1001 0000 1111的4变量more-or-less随机函数没有比

$$(\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge \bar{x}_4) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2) \tag{5}$$

更短的DNF表达式。该公式对应一条10步的布尔链。然而该函数也能更精巧地表达为

$$(((x_2 \wedge \bar{x}_4) \oplus \bar{x}_3) \wedge \bar{x}_1) \oplus x_2 \tag{6}$$

所以它的复杂度最多为4。

怎样发现（6）这样并非显而易见的公式呢？我们将会看到一台计算机无需进行大量的运算就可以找到4变量函数的最佳链。并且，所得结果可能非常令人吃惊，甚至对具备相当布尔代数经验的人来说也是如此。这种现象的典型例子可以在图9中看到，图中阐释了一些也许是最有意义的4变量函数，这些函数的变量在所有排列下都保持对称。

例如考虑函数$S_2(x_1, x_2, x_3, x_4)$，根据图9我们有

$$\begin{array}{rl}
x_1 & 0000\ 0000\ 1111\ 1111 \\
x_2 & 0000\ 1111\ 0000\ 1111 \\
x_3 & 0011\ 0011\ 0011\ 0011 \\
x_4 & 0101\ 0101\ 0101\ 0101 \\
x_5 = x_1 \oplus x_3 & 0011\ 0011\ 1100\ 1100 \\
x_6 = x_1 \oplus x_2 & 0000\ 1111\ 1111\ 0000 \\
x_7 = x_3 \oplus x_4 & 0110\ 0110\ 0110\ 0110 \\
x_8 = x_5 \vee x_6 & 0011\ 1111\ 1111\ 1100 \\
x_9 = x_6 \oplus x_7 & 0110\ 1001\ 1001\ 0110 \\
x_{10} = x_8 \wedge \bar{x}_9 & 0001\ 0110\ 0110\ 1000
\end{array} \tag{7}$$

这里给出真值表以便我们能轻易验证演算的每一步。步骤x_8产生一个函数，当$x_1 \neq x_2$或$x_1 \neq x_3$时为真，而$x_9 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$是奇偶校验函数$(x_1 + x_2 + x_1 + x_2) \bmod 2$。因此最终结果$x_{10}$为真当且仅当$\{x_1, x_2, x_3, x_4\}$中有两个为1；这些是满足$x_8$且奇偶校验为偶的情况。

图9的一些其他计算方案也能直观地予以判断。但是某些链，如对应$S_{1,4}$的那个，是相当有趣的。

注意，中间结果x_6在（7）中用了两次。实际上，对于函数$S_2(x_1, x_2, x_3, x_4)$，没有两次利用某些中间子表达式的6步链是不存在的；S_2的最短的代数公式包括如下般漂亮的对称性：

$$((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \oplus ((x_1 \vee x_2) \wedge (x_3 \vee x_4)) \tag{8}$$

它们都具有代价7。但图9表明其他4变量对称函数都能使用“纯”二叉树，即除外部结点（表示变量）外没有重叠的子树，最佳地进行求值。

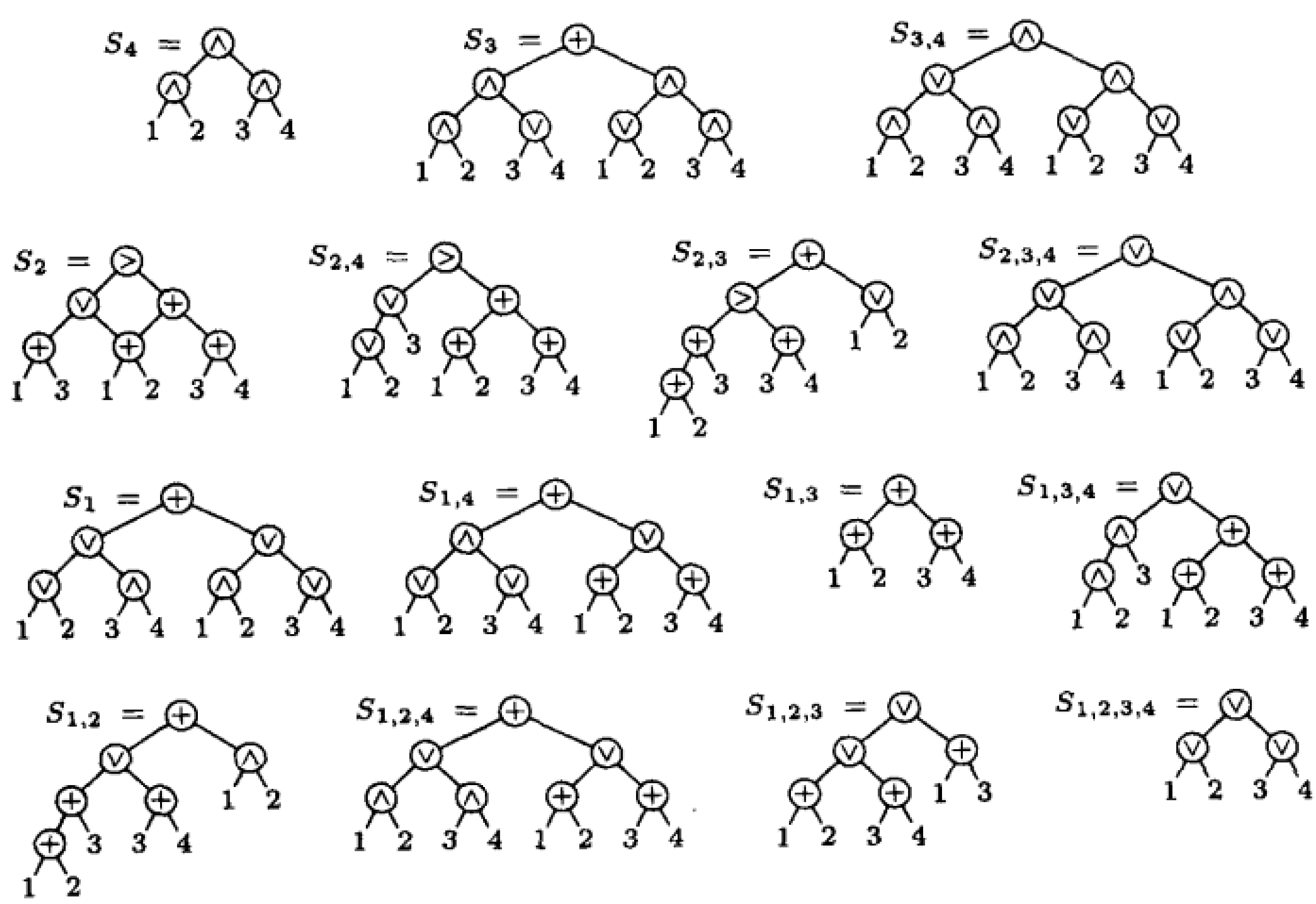


图9 4变量对称函数的最佳布尔链

一般来说，如果$f(x_1, \cdots, x_n)$是任意的布尔函数，我们称它的长度（length）$L(f)$是f的最短公式中二元操作符的数目。显然$L(f) \geqslant C(f)$；并且，通过考察7.1.1节(95)中3变量函数的14种基本类型，我们容易验证当$n \leqslant 3$时$L(f) = C(f)$。但我们刚刚看到当n=4时，$L(S_2) = 7$超过了$C(S_2) = 6$，而事实上，当n很大时，$L(f)$几乎总是远远大于$C(f)$（参见习题49）。

布尔函数f的深度（depth）$D(f)$是函数固有复杂度的另一个重要度量指标：我们称布尔链的深度是其对应树的最长下行路径的长度，而$D(f)$是考虑了f的所有布尔链后所得到的最小的深度。除了$S_{2,3}$和$S_{1,2}$两种情况，我们无法同时得到代价6和深度3外，图9所示的所有链不仅具有最小代价而且具有最小深度。公式

$$S_{2,3}(x_1,x_2,x_3,x_4)=((x_1\wedge x_2)\oplus(x_3\wedge x_4))\vee((x_1\vee x_2)\wedge(x_3\oplus x_4)) \qquad (9)$$

表明$D(S_{2,3})=3$，类似的公式也适用于$S_{1,2}$。

***n* = 4时的最佳链** 穷尽4变量函数的计算是可行的，因为这种函数只有2^{16} = 65 536种可能的真值表。事实上我们只需要考虑那些真值表的一半，因为任何函数f的补$\bar{f}$同f本身有着相同的代价、长度和深度。

若$f(0,\cdots,0)=0$，我们称$f(x_1,\cdots,x_n)$是规范的（normal），一般来说

$$f(x_1,\cdots,x_n)\oplus f(0,\cdots,0) \qquad (10)$$

是f的“范式化”（normalization）。任何布尔链能通过对它的每一步进行范式化和对其操作符作适当变化，从而成为规范的；因为若$(\hat{x}_1,\cdots,\hat{x}_{i-1})$是$(x_1,\cdots,x_{i-1})$的范式化且如（1）那样有$x_i=x_{j(i)}\circ_i x_{k(i)}$，那么$\hat{x}_i$显然是$\hat{x}_{j(i)}$和$\hat{x}_{k(i)}$的二元函数（习题7给出一个例子）。因此，我们可将考虑限制在规范的布尔链上，而不失一般性。

注意，一条布尔链是规范的当且仅当它的每个二元运算符$\circ_i$均是规范的。而规范的二元运算符只有8种，它们中的3种即⊥、L和R是不重要的。所以可假定所有我们感兴趣的布尔链是由5种操作符∧、⊂、⊃、∨和⊕构成的，在图9中分别用Ⓐ、⊂○、⊃○、Ⓥ和⊕表示。此外我们可假定每一步有$j(i)<k(i)$。

共有2^{15} = 32 768个4变量的规范函数，而且我们能通过有系统地枚举出所有长度为0、1、2等的函数轻易计算它们的长度。事实上，$L(f)=r$意味着对某些g和h，$f=g\circ h$，其中$L(g)+L(h)=r-1$且$\circ$是5个重要的规范运算符之一；所以我们能进行如下计算过程。

算法L（寻找规范长度，find normal length） 该算法对所有规范真值表$0\leqslant f<2^{2^n-1}$，通过建立所有长度为$r(r\geqslant 0)$的非零规范函数列表确定$L(f)$。

L1. [初始化。] 令$L(0)\leftarrow 0$和$L(f)\leftarrow\infty$对$1\leqslant f<2^{2^n-1}$。接着，对$1\leqslant k\leqslant n$，令$L(x_k)\leftarrow 0$并将x_k放入列表0中，这里

$$x_k=(2^{2^n}-1)/(2^{2^{n-k}}+1) \qquad (11)$$

是x_k的真值表（参见习题8）。最后，令$c\leftarrow 2^{2^n-1}-n-1$；$c$是$L(f)=\infty$位置的数目。

L2. [对r进行循环] 对$r=1,2,\cdots$执行L3；最终算法在c为0时终止。

L3. [对j和k进行循环] 对$j=0,1,\cdots$，$k=r-1-j$在$j\leqslant k$时执行L4。

L4. [对g和h进行循环] 对列表j中的所有g和列表k中的所有h执行L5。（若$j=k$，将h限制为列表k中g随后的函数就足够了。）

L5. [对f进行循环] 对$f=g\ \&\ h$，$f=\bar{g}\ \&\ h$，$f=g\ \&\ \bar{h}$，$f=g\mid h$和$f=g\oplus h$执行L6。（这里$g\ \&\ h$表示整数g和h的按位AND；我们用整数的二进制记法表示真值表。）

L6. [f是新的？] 若$L(f)=\infty$，令$L(f)\leftarrow r$，$c\leftarrow c-1$，并将f放入列表r中。若$c=0$则终止算法。▌

习题10给出了一个计算所有深度$D(f)$的类似过程。

稍微多做一点工作，通过计算一个被称为f的“足迹”（footprint）的启发式位向量$\phi(f)$，实际上我们就能修改算法L以使它能找到更好的$C(f)$的上界。一条规范

的布尔链只能以$5\binom{n}{2}$种不同的方式开始，因为第一步x_{n+1}必须是$x_1 \wedge x_2$或$\bar{x}_1 \wedge x_2$或$x_1 \wedge \bar{x}_2$或$x_1 \vee x_2$或$x_1 \oplus x_2$或$x_1 \wedge x_3$或……或$x_{n-1} \oplus x_n$中的一种。假设$\phi(f)$是一个长度为$5\binom{n}{2}$的位向量而$U(f)$是$C(f)$的一个上界，它们具有如下性质：$\phi(f)$的每1位相应于某条在$U(f)$步内计算f的布尔链的第一步。

表1 给定复杂度的4变量函数的数目

$C(f)$	类	函数	$L(f)$	类	函数	$D(f)$	类	函数
0	2	10	0	2	10	0	2	10
1	2	60	1	2	60	1	2	60
2	5	456	2	5	456	2	17	1 458
3	20	2 474	3	20	2 474	3	179	56 456
4	34	10 624	4	34	10 624	4	22	7 552
5	75	24 184	5	75	24 184	5	0	0
6	72	25 008	6	68	24 640	6	0	0
7	12	2 720	7	16	3 088	7	0	0

这样的对$(U(f), \phi(f))$可通过对算法L的基本策略进行扩展而计算得到。初始时令$U(f) \leftarrow 1$并且对所有代价为1的函数f，将$\phi(f)$设置为适当的向量0…010…0。接着，对$r = 2, 3, \cdots$，寻找函数$f = g \circ h$，这里$U(g) + U(h) = r - 1$，过程和前面一样，但有两处改动：（1）若g和h的足迹至少有一个共同的元素，即$\phi(g) \mathbin{\&} \phi(f) \neq 0$，那么可知$C(f) \leqslant r - 1$，因此如果$U(f) \geqslant r$就可减小$U(f)$的值。（2）若$g \circ h$的代价等于（但不小于）当前的上界$U(f)$，若$U(f) = r$，则可令$\phi(f) \leftarrow \phi(f) \mid (\phi(g) \mid \phi(h))$；若$U(f) = r - 1$，则可令$\phi(f) \leftarrow \phi(f) \mid (\phi(g) \mathbin{\&} \phi(h))$。细节参见习题11。

结果证明这个基于足迹的启发式方法能力很强，当n=4时，它可以发现所有函数f的最佳代价$U(f) = C(f)$的链。此外，不久也将看到使用足迹也能帮助我们解决更复杂的计算问题。

根据7.1.1节表5，当我们忽略由变量的排列或值的互补造成的微小差别时，2^{16} = 65 536个4变量函数仅属于222个不同的类。从算法L和其变体可得表1所示的汇总统计数据。

***使用最少的内存进行计算** 假定布尔值$x_1, \cdots, x_n$出现在n个寄存器中，而我们想通过执行一个操作序列来计算一个函数，其中操作的形式为

$$x_{j(i)} \leftarrow x_{j(i)} \circ_i x_{k(i)} \quad 对于 1 \leqslant i \leqslant r \tag{12}$$

这里$1 \leqslant j(i) \leqslant n$，$1 \leqslant k(i) \leqslant n$，且$\circ_i$是二元操作符。在计算的最后，需求的函数值应该出现在某个寄存器中，如当n=3时，下面4个步骤的序列

$$\begin{array}{ll} & (x_1 = 00001111 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_1 \leftarrow x_1 \oplus x_2 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_3 \leftarrow x_3 \wedge x_1 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 00010100) \\ x_2 \leftarrow x_2 \wedge \bar{x}_1 & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010100) \\ x_3 \leftarrow x_3 \vee x_2 & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010111) \end{array} \tag{13}$$

计算中数<$x_1\ x_2\ x_3$>并将其置于x_3原来的寄存器中（每一步操作前后，寄存器内容的所有8种可能性都以真值表形式在这里呈现）。

实际上，我们可以通过每次只使用一个真值表，而非保持全部三个真值表的径迹来检查计算过程。如果我们对情景进行逆向分析，令$f_l(x_1, \cdots, x_n)$表示由序列的$l, l+1, \cdots, r$步计算出的函数，省略前$l-1$步；即，在我们的例子中，$f_2(x_1, x_2, x_3)$将是进行了3步计算$x_3 \leftarrow x_3 \wedge x_1$，$x_2 \leftarrow x_2 \wedge \bar{x}_1$，$x_3 \leftarrow x_3 \vee x_2$后在$x_3$中的结果。然后，寄存器$x_3$中存放的由所有4步计算出的函数是

$$f_1(x_1, x_2, x_3) = f_2(x_1 \oplus x_2, x_2, x_3) \tag{14}$$

类似地，$f_2(x_1, x_2, x_3) = f_3(x_1, x_2, x_3 \wedge x_1)$，$f_3(x_1, x_2, x_3) = f_4(x_1, x_2 \wedge \bar{x}_1, x_3)$，$f_4(x_1, x_2, x_3) = f_5(x_1, x_2, x_3 \vee x_2)$，以及$f_5(x_1, x_2, x_3) = x_3$。因此我们能通过对真值表进行适当的操作，而从$f_5$到$f_4$到……到$f_1$返回。

例如，假定$f(x_1, x_2, x_3)$是一个函数，它的真值表是

$$t = a_0a_1a_2a_3a_4a_5a_6a_7$$

那么$g(x_1, x_2, x_3) = f(x_1 \oplus x_2, x_2, x_3)$的真值表是

$$u = a_0a_1a_6a_7a_4a_5a_2a_3$$

它由$a_{x'}$替换a_x得到，其中

$$x = (x_1x_2x_3)_2 \text{ 蕴涵着 } x' = ((x_1 \oplus x_2)x_2x_3)_2$$

类似地，比如说$h(x_1, x_2, x_3) = f(x_1, x_2, x_3 \wedge x_1)$的真值表是

$$v = a_0a_0a_2a_2a_4a_5a_6a_7$$

并且能使用按位操作从t计算u和v：

$$u = t \oplus ((t \oplus (t \gg 4) \oplus (t \ll 4)) \& (00110011)_2) \tag{15}$$

$$v = t \oplus ((t \oplus (t \gg 1)) \& (01010000)_2) \tag{16}$$

令$C_m(f)$是一个最短的使用最少内存的f的计算长度。逆向计算原理告诉我们，如果我们知道$C_m(f) < r$的所有函数f的真值表，我们能很快找到$C_m(f) = r$的所有函数的真值表。即我们能像前面一样仅考虑规范函数。那么，对所有使$C_m(g) = r - 1$的规范的g，对

$$g(x_1, \cdots, x_{j-1}, x_j \circ x_k, x_{j+1}, \cdots, x_n) \tag{17}$$

我们可构造$5n(n-1)$个真值表，如果它们之前没有被标记，就用代价r来标记它们。习题14说明那些真值表能通过在g的真值表上执行简单的按位运算而计算出来。

当$n = 4$时，所有222个基本函数类型中除了13个以外均被证明有$C_m(f) = C(f)$，因此，它们均能使用最少的内存进行计算而不增加代价。特别是，所有对称函数都具有这个性质，尽管从图9来看这个事实一点也不明显。有5类函数有$C(f) = 5$而$C_m(f) = 6$；有8类函数有$C(f) = 6$但$C_m(f) = 7$。后一种类型最有趣的例子大概是函数$(x_1 \vee x_2) \oplus (x_3 \vee x_4) \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$，由公式

$$x_1 \oplus (x_3 \vee x_4) \oplus (x_2 \wedge (\bar{x}_1 \vee (x_3 \wedge x_4))) \tag{18}$$

它有代价6，但它没有长度小于7的最少内存链（参见习题15）。

***确定最小代价** 通过观察所有明显符合下面三种情况中至少一种的f的最佳布尔链$(x_{n+1}, \cdots, x_{n+r})$就可找到$C(f)$的确切值：

i) $x_{n+r} = x_j \circ x_k$，其中x_j和x_k没有共同的中间结果；

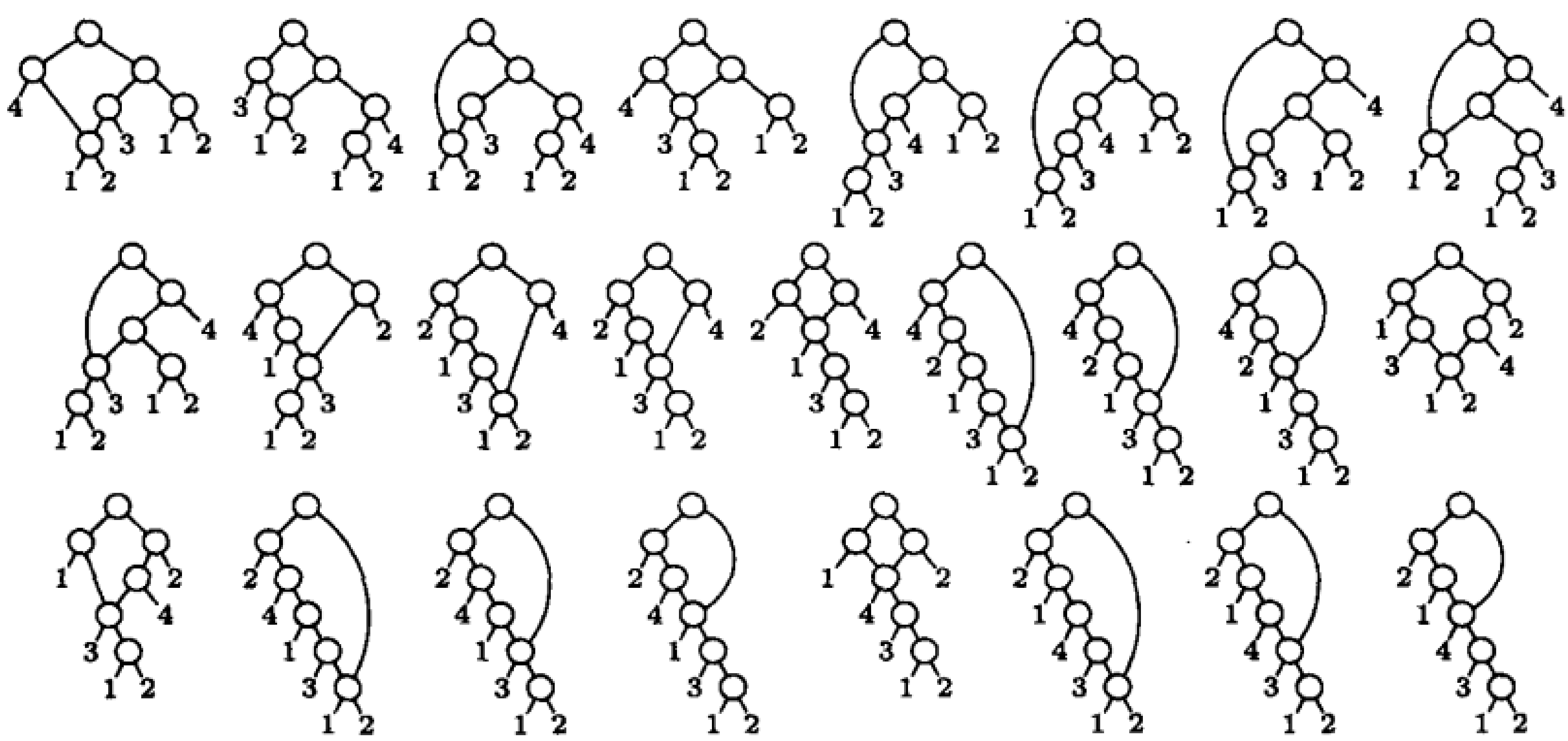
ii) $x_{n+1} = x_j \circ x_k$，其中x_j和x_k都没有用在$x_{n+2}, \cdots, x_{n+r}$步；

iii) 以上都不是，即便中间步骤进行重新编号也是如此。

在情况(i)中我们有$f = g \circ h$，其中$C(g) + C(h) = r - 1$，并且我们称之为“自顶向下”构造。在情况(ii)中我们有$f(x_1, \cdots, x_n) = g(x_1, \cdots, x_{j-1}, x_j \circ x_k, x_{j+1}, \cdots, x_n)$，其中$C(g) = r - 1$；我们称之为“自底向上”构造。

仅递归使用自顶向下构造的最佳链对应最小公式长度$L(f)$。仅递归使用自底向上构造的最佳链对应使用最小内存的计算，长度为$C_m(f)$。通过混合使用自顶向下构造和自底向上构造，我们还能做得更好；但仍无法知道是否找到了$C(f)$，因为一条属于情况(iii)的特殊链可能更短。

幸运的是，这种特殊链很少，因为它们必须满足相当强的条件，而且当n和r不太大时它们能被穷举。例如，习题19证明了当$r < n+2$时不存在特殊链，而当$n=4$，$r=6$时，只有25条本质上不同的特殊链，但显然不是长度更短的链。



通过系统地尝试每条特殊链的5^r种可能性，每种可能性对应着对树的内部结点赋予规范运算符的方式，我们能在每个等价类中找到至少一个函数f，其最小代价$C(f)$只在情况(iii)得到。

实际上，当$n = 4$和$r = 6$时，这$25 \cdot 5^6 = 390\ 625$次尝试只产生一类函数，它不能通过任何自顶向下加自底向上链在6步内计算。缺少的类以部分对称函数$(\langle x_1x_2x_3\rangle \vee x_4) \oplus (x_1 \wedge x_2 \wedge x_3)$为典型，可通过适当地对上面所描述的前5条链特化而在6步内达到；例如，一种方式是

$$x_5 = x_1 \wedge x_2,\ x_6 = x_1 \vee x_2,\ x_7 = x_3 \oplus x_5,\quad x_8 = x_4 \wedge \bar{x}_5,\ x_9 = x_6 \wedge x_7,\ x_{10} = x_8 \vee x_9 \tag{19}$$

对应第一条特殊链。因为所有其他函数有$L(f) \leqslant 7$，这些尝试性计算已经确立了所

有情况下的真实的最小代价。

历史注释：《Annals of the Computation Laboratory of Harvard University》27 (1951) 首次报告了以最优方式计算所有布尔函数$f(w,x,y,z)$的合作尝试，其中Howard Aiken的职员提出了启发式方法和他们能够构造的最好的变换电路的扩展表。他们的代价度量$V(f)$和这里考虑的$C(f)$不同，它是基于真空电子管“控制栅格”的。它们有3种门，NOR、OR和NAND，每种能接受k个输入，代价为k。门的每个输入既可以是变量，也可以是变量的补或前一个门的输出。此外，计算的函数作为任意数量门的AND出现在最高层，它没有额外的代价。

基于这些代价标准，函数可能和它的补不具有相同的代价，原因是AND门只能在最高层。可以计算$x\wedge y$作为NOR($\bar{x}$, $\bar{y}$)，有代价2；但$\bar{x}\vee(\bar{y}\wedge\bar{z})$ = NAND(x, OR(y,z))的代价是4，而它的补$x\wedge(y\vee z)$ = AND(NOR($\bar{x}$), OR(y,z))的代价只有3。因此Harvard的研究者需要考虑402类本质不同的4变量函数，而不是222类（参见7.1.1节习题125的答案）。当然，那时候他们是纯手工工作的。他们发现$V(f)<20$的所有情况，除了64个等价于$S_{0,1}(w,x,y,z)\vee(S_2(w,x,y)\wedge z)$的函数外，他们用20个控制栅格计算如下函数：

$$g_1=\mathrm{NOR}(\bar{w},\bar{x}),\ g_2=\mathrm{NAND}(\bar{y},z),\ g_3=\mathrm{NOR}(w,x),$$
$$f=\mathrm{AND}(\mathrm{NAND}(g_1,g_2),\mathrm{NAND}(g_3,\mathrm{NOR}(\bar{y},\bar{z})),\quad\mathrm{NOR}(\mathrm{NOR}(g_3,\bar{y},z),\mathrm{NOR}(g_1,g_2,g_3)))\tag{20}$$

Leo Hellerman编写了第一个可找出可证最优电路的计算机程序[《IEEE Transactions》EC-12 (1963), 198—223]，对于任何的给定函数$f(x,y,z)$，程序确定了计算该函数所需的NOR门的最少数目。他要求每个门的输入要么是非补变量，要么是前面门的输出；扇入和扇出数限定最多为3。当两个电路具有相同的门数，他选择输入数之和最小的那个。如，计算$\bar{x}$ = NOR(x)的代价为1；计算$x\vee y\vee z$ = NOR(NOR(x,y,z))的代价为2；$\langle xyz\rangle$ = NOR(NOR(x,y), NOR(x,z), NOR(y,z)) 代价为4；$S_1(x,y,z)$ = NOR(NOR(x,y,z), $\langle xyz\rangle$)代价为6等。因为他将扇出限定在3，他发现每个3变量函数能以代价7或更少予以计算，除了奇偶校验函数$x\oplus y\oplus z=(x\equiv y)\equiv z$，其中$x\equiv y$代价为4，因为它是NOR(NOR($x$, NOR($x,y$)), NOR($y$, NOR($x,y$)))。

表2 给定复杂度的5变量函数的数目

$C(f)$	类	函数	$L(f)$	类	函数	$D(f)$	类	函数
0	2	12	0	2	12	0	2	12
1	2	100	1	2	100	1	2	100
2	5	1 140	2	5	1 140	2	17	5 350
3	20	11 570	3	20	11 570	3	1 789	6 702 242
4	93	109 826	4	93	109 826	4	614 316	4 288 259 592
5	389	995 240	5	366	936 440	5	0	0
6	1 988	8 430 800	6	1 730	7 236 880	6	0	0
7	11 382	63 401 728	7	8 782	47 739 088	7	0	0
8	60 713	383 877 392	8	40 297	250 674 320	8	0	0
9	221 541	1 519 125 536	9	141 422	955 812 256	9	0	0

（续）

$C(f)$	类	函数	$L(f)$	类	函数	$D(f)$	类	函数
10	293 455	2 123 645 248	10	273 277	1 945 383 936	10	0	0
11	26 535	195 366 784	11	145 707	1 055 912 608	11	0	0
12	1	1 920	12	4 423	31 149 120	12	0	0

电气工程师继续探索其他的代价标准，但4变量函数看上去遥不可及，直到1977年，Frank M. Liang建立了表1所示的$C(f)$值。Liang未发表的推导是基于对所有不能归约为自底向上构造的链进行研究的结果的。

n = 5的情况 根据7.1.1节表5，共有616 126类本质不同的函数$f(x_1, x_2, x_3, x_4, x_5)$。现在计算机已经足够快，这个数目已不再吓人；所以作者在写这一节的时候决定研究所有5变量布尔函数的$C(f)$。幸好有点运气，确实能得到完整的结果，从而得到表2所示的统计数据。

对此计算，对算法L和它的变体进行修改以处理类的代表，而不是用由全部2^{31}个规范真值表组成的集合。使用7.2.1.2节习题20中的方法可使生成一个类的所有函数更为简单，给定它们中的任一个，所产生的加速都是千倍以上的。对自底向上方法略微加强，使得其在$C(f) = r - 2$时能推出例如$f(x_1 \wedge x_2, x_1 \vee x_2, x_3, x_4, x_5)$有代价$\leq r$这样的结果。找到所有代价为10的类后，自顶向下和自底向上的方法能找到除了7类函数以外所有长度≤ 11的链。然后费时的部分开始了，这里生成了大约5300万条n=5和r=11的特殊链；每条这样的链导向5^{11} = 48 828 125个函数，其中某些很可能落入剩下的7个神秘的类中。但其中只有6类被发现是有11步解决方案的。那个孤单的幸存者，其16进制表示的真值表为`169ae443`，是个独特的类，其$C(f)$ = 12，并且它也有$L(f)$ = 12。

图10展示了所产生的对称函数的结构，它们中的一些惊人地优美，一些美得简单，而另一些则简单得惊人。（例如，看一下8步计算$S_{2,3}(x_1, x_2, x_3, x_4, x_5)$或$S_{2,3,4}$优雅的公式，或$S_{4,5}$和$S_{3,4,5}$的非单调链。）顺便提一句，表2表明所有5变量函数具有深度≤ 4，但图10没有尝试对深度进行最小化。

结果表明，所有这些对称函数能使用最少的内存进行计算而无需增加代价。但这个事实目前还没有简单的证明。

多重输出 我们常要计算一些具有相同输入值$x_1, \cdots, x_n$的不同布尔函数$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$，换句话说，我们经常想要计算多路函数$y = f(x)$，其中$y = f_1 \cdots f_m$是长度为$m$的二进制向量且$x = x_1 \cdots x_n$是长度为$n$的二进制向量。幸运的是，计算一个组件值$f_j(x_1, \cdots, x_n)$的许多工作可以同计算其他组件值$f_k(x_1, \cdots, x_n)$所需要的运算共享。

令$C(f) = C(f_1 \cdots f_m)$为计算所有重要函数f_j的一条最短布尔链的长度。更精确地，链$(x_{n+1}, \cdots, x_{n+r})$必须具有如下性质：对$1 \leq j \leq m$，要么$f_j(x_1, \cdots, x_n) = x_{l(j)}$，要么$f_j(x_1, \cdots, x_n) = \bar{x}_{l(j)}$，对某个$l(j)$有$0 \leq l(j) \leq n+r$，其中$x_0 = 0$。显然$C(f) \leq C(f_1) + \cdots + C(f_m)$，但我们也许能做得更好。

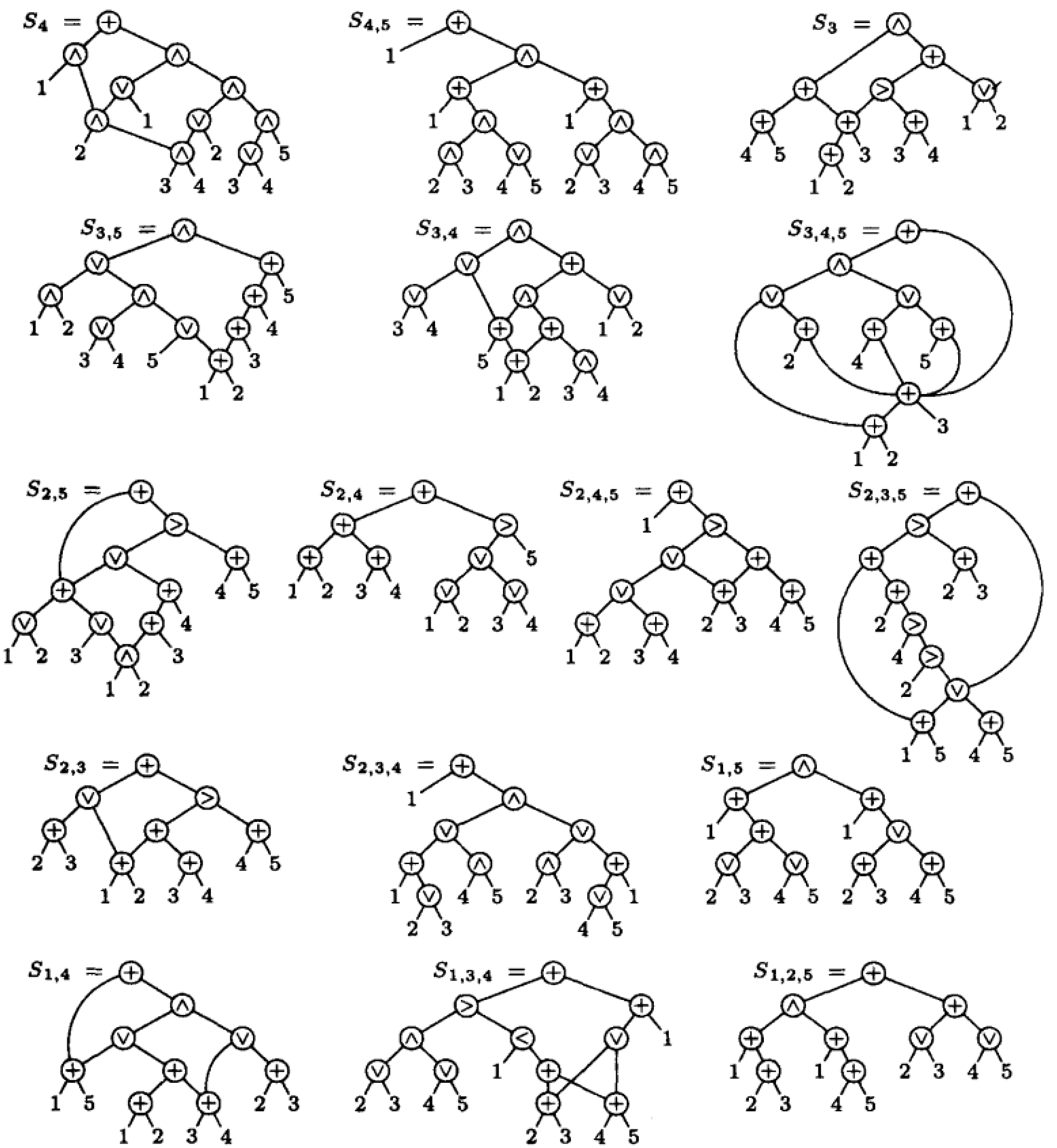


图10 5变量对称函数的最小代价布尔链

例如，假定我们想要计算函数z_1和z_0，由

$$(z_1z_0)_2 = x_1 + x_2 + x_3 \tag{21}$$

定义的3个布尔变量的两位二进制和，我们有

$$z_1 = \langle x_1x_2x_3\rangle \quad 和 \quad z_0=x_1\oplus x_2\oplus x_3 \tag{22}$$

所以独立的代价是$C(z_1) = 4$和$C(z_0) = 2$。但容易见到组合代价$C(z_1z_0)$最多是5，因为$x_1 \oplus x_2$在z_j的每一位计算中是合适的第一步。

$$\begin{aligned} &x_4 = x_1 \oplus x_2, \quad z_0 = x_5 = x_3 \oplus x_4 \\ &x_6 = x_3 \wedge x_4, \quad x_7 = x_1 \wedge x_2, \quad z_1 = x_8 = x_6 \vee x_7 \end{aligned} \tag{23}$$

此外，穷尽计算表明$C(z_1z_0) > 4$；因此$C(z_1z_0) = 5$。

电气工程师通常称（21）的电路为全加器（full adder），因为n个这样的结构

单元可以钩接在一起以完成两个n位二进制数相加的任务。(22) 中$x_3 = 0$的特例也很重要，其可简化为

$$z_1 = x_1 \wedge x_2 \quad 和 \quad z_0 = x_1 \oplus x_2 \tag{24}$$

且有复杂度2；工程师称它为“半加器”(half adder)，注意，全加器的代价超过了两个半加器的代价。

以2为基的加法的普遍问题

$$\begin{array}{r} (x_{n-1}\cdots x_1x_0)_2 \\ \underline{(y_{n-1}\cdots y_1y_0)_2} \\ (z_nz_{n-1}\cdots z_1z_0)_2 \end{array} \tag{25}$$

是由$2n$个布尔输入$x_{n-1}\cdots x_1x_0y_{n-1}\cdots y_1y_0$计算$n+1$个布尔输出$z_n\cdots z_1z_0$的问题；这马上可通过下述公式解决。

$$c_{j+1} = \langle x_jy_jc_j\rangle, z_j = x_j \oplus y_j \oplus c_j, 对于0 \leqslant j < n \tag{26}$$

其中c_j是“进位”且有$c_0 = 0$，$z_n = c_n$。因此我们能用半加器计算c_1和z_0，跟随其后的是计算其他c和z的$n-1$个全加器，其累积总代价为$5n - 3$。实际上，N. P. Red'kin [《Problemy Kibernetiki》38 (1981), 181—216] 通过35页的详尽归纳证明，推断了情况2.2.2.3.1.2.3.2.4.3(!)，证实了确实$5n-3$步的计算是必须的。但其电路的深度为$2n-1$，其对实际的并行计算来说太大了，所以很多学者致力于研究深度为$O(\log n)$，且有合理代价的加法电路的设计工作（参见习题41—44）。

现在让我们扩展 (21) 并尝试计算一般的“横向之和”

$$(z_{\lfloor \lg n\rfloor}\cdots z_1z_0)_2 = x_1 + x_2 + \cdots + x_n \tag{27}$$

若$n = 2k + 1$，我们能用k个全加器将和约简为$(x_1+\cdots+x_n) \bmod 2$加上k个权重为2的位，原因是每个全加器都减少权重数目，1位的权重数目减少2。例如，若$n=9$且$k=4$的计算是

$$x_{10} = x_1 \oplus x_2 \oplus x_3,\ x_{11} = x_4 \oplus x_5 \oplus x_6,\ x_{12} = x_7 \oplus x_8 \oplus x_9,\ x_{13} = x_{10} \oplus x_{11} \oplus x_{12}$$
$$y_1 = \langle x_1x_2x_3\rangle,\ y_2 = \langle x_4x_5x_6\rangle,\ y_3 = \langle x_7x_8x_9\rangle,\ y_4 = \langle x_{10}x_{11}x_{12}\rangle$$

而我们有$x_1+\cdots+x_9 = x_{13} + 2(y_1+y_2+y_3+y_4)$。若$n=2k$是偶数，类似的约简同样适用，但最后采用的是半加器。这样权重为2的位能够以相同的方式相加；所以我们得到计算$z_{\lfloor \lg n\rfloor}\cdots z_1z_0$所需要的门的总数的递推关系

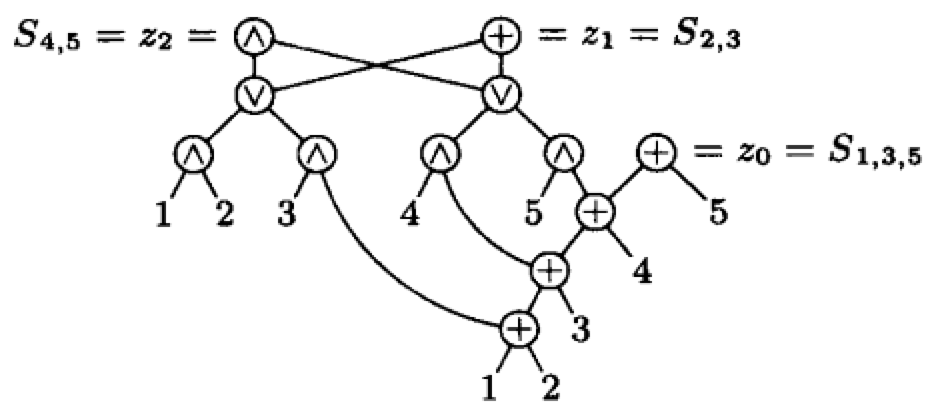
$$s(n) = 5\lfloor n/2\rfloor - 3[n \text{ even}] + s(\lfloor n/2\rfloor),\quad s(0) = 0 \tag{28}$$

($s(n)$的一个相近的公式出现在习题30中)。我们有$s(n) < 5n$，且开头的值

n=1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

$s(n)$=0 2 5 9 12 17 20 26 29 34 37 44 47 52 55 63 66 71 74 81

表明该方法甚至对较小的n也相当高效。例如，$n=5$，其生成的是



(29)

它在正好12步内计算3个不同的对称函数$z_2 = S_{4,5}(x_1, \cdots, x_5)$，$z_1 = S_{2,3}(x_1, \cdots, x_5)$，$z_0 = S_{1,3,5}(x_1, \cdots, x_5)$。根据图10，$S_{4,5}$的10步计算是最优化的；当然$S_{1,3,5}$的4步计算也是最优化的。此外，虽然$C(S_{2,3}) = 8$，这里以一种巧妙的10步方式，与$S_{4,5}$共享除了一个门之外的所有门来计算函数$S_{2,3}$。

注意，我们现在能高效地计算任何对称函数，因为$\{x_1, \cdots, x_n\}$的每个对称函数是$z_{\lfloor \lg n \rfloor} \cdots z_1 z_0$的一个布尔函数。例如，我们知道任何4变量布尔函数具有复杂度$\leqslant 7$；因此，任何对称函数$S_{k_1,\cdots,k_t}(x_1, \cdots, x_{15})$的代价最多为$s(15) + 7 = 62$。令人吃惊的是：当$n$较小时，$n$变量对称函数是所有里面最难计算的之一，但当$n \geqslant 10$时，它们是最容易计算的之一。

我们也能高效地计算对称函数的集合。即如果我们要计算所有$n+1$个对称函数$S_k(x_1, \cdots, x_n)$，对$0 \leqslant k \leqslant n$有单独一条布尔链，我们只需计算头$n+1$个极小项（minterm）$z_0$，$z_1$，$\cdots$，$z_{\lfloor \lg n \rfloor}$。例如，当$n=5$时那些给予我们所有函数$S_k$的极小项分别为$S_0 = \bar{z}_0 \wedge \bar{z}_1 \wedge \bar{z}_2$，$S_1 = \bar{z}_0 \wedge \bar{z}_1 \wedge z_2$，$\cdots$，$S_5 = z_0 \wedge \bar{z}_1 \wedge z_2$。

计算n个变量的所有2^n个极小项有多难呢？电气工程师称这个函数为n到2^n二进制编码器（binary decoder），因为它将n位$x_1 \cdots x_n$转换到2^n位的序列$d_0 d_1 \cdots d_{2^n-1}$，其中恰有一个为1。"分而治之"的原理建议我们先计算所有前$\lceil n/2 \rceil$个变量的极小项，以及所有后$\lfloor n/2 \rfloor$个变量的极小项；然后使用2^n个AND门即可完成工作。这种方法的代价是$t(n)$，其中

$$t(0) = t(1) = 0; \quad t(n) = 2^n + t(\lceil n/2 \rceil) + t(\lfloor n/2 \rfloor) \text{ 当} n \geqslant 2 \tag{30}$$

所以$t(n) = 2^n + O(2^{n/2})$；大体上每个极小项一个门（参见习题32）。

具有多重输出的函数往往能帮助我们建立更大的具有单个输出的函数。例如，我们已经看到横向加法器（27）允许我们计算对称函数；而n到2^n编码器也有很多应用，尽管当n较大时2^n会非常巨大。一种适用的情况是2^m路多路器$M_m(x_1, \cdots, x_m; y_0, \cdots, y_{2^m-1})$，同样被称为$m$位存储访问函数，它具有$n = m + 2^m$个输入，当$(x_1, \cdots, x_m)_2 = k$时，取值$y_k$。由定义我们有

$$M_m(x_1, \cdots, x_m;\ y_0, y_1, \cdots, y_{2^m-1}) = \bigvee_{k=0}^{2^m-1} (d_k \wedge y_k) \tag{31}$$

其中d_k是m到$2m$二进制编码器的第k个输出；因此，由（30）我们能用$2^m + (2^m - 1) + t(m) = 3n + O(\sqrt{n})$个门计算$M_m$。但习题39表明我们确实能将代价减至只有$2n + O(\sqrt{n})$（同样可参见习题79）。

渐近事实 当变量的数目小时，我们的穷尽搜索方法已经找到大量案例，其中布尔函数能以令人眩晕的效率计算。所以自然期望，当更多变量出现时，更多精巧的计算也会出现。但事实恰恰相反，至少从统计学观点来看是这样的。

S定理 *几乎每个布尔函数$f(x_1, \cdots, x_n)$的代价皆超过$2^n/n$。更精确地说，若$c(n, r)$个布尔函数有复杂度$\leqslant r$，则有*

$$(r-1)!c(n, r) \leqslant 2^{2r+1}(n+r-1)^{2r} \qquad (32)$$

证明：若一个函数能在$r-1$步内计算，它也能由一条r步的链计算。（该陈述当$r=1$时是显然的；否则我们能令$x_{n+r}=x_{n+r-1} \wedge x_{n+r-1}$。）我们将说明$r$步的链不会很多，因此我们不能计算很多代价$\leqslant r$的不同函数。

令π为$\{1, \cdots, n+r\}$的一个排列，取$1 \mapsto 1, \cdots, n \mapsto n$，且$n+r \mapsto n+r$；共有$(r-1)!$个这样的排列。假定$(x_{n+1}, \cdots, x_{n+r})$是一条布尔链，其中每个中间步骤$x_{n+1}, \cdots, x_{n+r-1}$至少在一个后续步骤中使用一次。那么由规则

$$x_i = x_{j'(i)} \circ_i' x_{k'(i)} = x_{j(i\pi)\pi^-} \circ i\pi x_{k(i\pi)\pi^-}, \text{对于} n<i\leqslant n+r \qquad (33)$$

定义的改序的（permuted）链对不同的π是不同的（若π取$a \mapsto b$，我们写作$b=a\pi$且$a=b\pi^-$）。例如，若π取$5 \mapsto 6 \mapsto 7 \mapsto 8 \mapsto 9 \mapsto 5$，链（7）变成

原始的	改序的
$x_5 = x_1 \oplus x_3$	$x_5 = x_1 \oplus x_2$
$x_6 = x_1 \oplus x_2$	$x_6 = x_3 \oplus x_4$
$x_7 = x_3 \oplus x_4$	$x_7 = x_9 \vee x_5$
$x_8 = x_5 \vee x_6$	$x_8 = x_5 \oplus x_6$
$x_9 = x_6 \oplus x_7$	$x_9 = x_1 \oplus x_3$
$x_{10} = x_8 \wedge \bar{x}_9$	$x_{10} = x_7 \wedge \bar{x}_8$

(34)

注意，与通常的规则相反，我们可能有$j'(i) \geqslant k'(i)$或$j'(i) > i$或$k'(i) > i$。但改序的链和以前一样计算同样的函数x_{n+r}，并且它没有任何回路，即一个入口间接定义它本身，因为改序的x_i是原来的$x_{i\pi}$。

我们能将考虑限制在之前谈到的规范（normal）布尔链。所以$c(n, r)/2$个代价$\leqslant r$的规范布尔函数导致$(r-1)!c(n, r)/2$条不同的改序的链，其中每一步中的运算符$\circ_i$是$\wedge$，$\vee$，$\supset$或$\oplus$中的一个。而最多有$4^r(n+r-1)^{2r}$条这样的链，因为对$\circ_i$有4种算法，而对每个$j(i)$和$k(i)$有$n+r-1$种选择，其中$n<i\leqslant n+r$。于是有等式（32）；而我们通过设$r=\lfloor 2^n/n \rfloor$，得到了定理开始的陈述（参见习题46）。▮

另一方面，对关注无限的人来说也有好的消息：采用C. E. Shannon的设计技术、O. B. Lupanov改进的技术［《Bell System Tech. J》28 (1949), 59—98, Theorem 6;《Isvestiia VUZov, Radiofizika》1 (1958), 120—140］，我们实际上能计算每个n变量布尔函数，但即使避开⊕和≡，也需要略微大于$2^n/n$步的计算。

事实上，Shannon-Lupanov方法即使n较小时也可导致有用的结果，所以让我们通过研究一个小的实例来了解它。考虑

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = [(x_1x_2x_3x_4x_5x_6)_2\text{是素数}] \qquad (35)$$

是一个识别所有6位素数的函数。它的真值表有$2^6 = 64$位，而我们能采用4×16的数组来看那些位，从而方便地与它共事，而不是把我们自己限制在一维世界中。

$$\begin{array}{r|cccccccccccccccc|l}
\multicolumn{1}{r}{x_3 =} & 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&\multicolumn{1}{c}{1} & \\
\multicolumn{1}{r}{x_4 =} & 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&\multicolumn{1}{c}{1} & \\
\multicolumn{1}{r}{x_5 =} & 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&\multicolumn{1}{c}{1} & \\
\multicolumn{1}{r}{x_6 =} & 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&\multicolumn{1}{c}{1} & \\
\cline{2-17}
x_1x_2 = 00 & 0&0&1&1&0&1&0&1&0&0&0&1&0&1&0&0 & \multirow{2}{*}{\}\text{组}1} \\
x_1x_2 = 01 & 0&1&0&1&0&0&0&1&0&0&0&0&0&1&0&1 & \\
x_1x_2 = 10 & 0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&1 & \multirow{2}{*}{\}\text{组}2} \\
x_1x_2 = 11 & 0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0 & \\
\cline{2-17}
\end{array} \qquad (36)$$

这些行被分为两组，每组两行；每行有16列，4种基本类型，即$\begin{smallmatrix}0\\0\end{smallmatrix}$，$\begin{smallmatrix}0\\1\end{smallmatrix}$，$\begin{smallmatrix}1\\0\end{smallmatrix}$或$\begin{smallmatrix}1\\1\end{smallmatrix}$。因此我们看到函数能表达为

$$\begin{aligned}
f(x_1,\ldots,x_6) \;=\; & \quad([x_1x_2 \in \{00\}] \quad\;\; \wedge [x_3x_4x_5x_6 \in \{0010,0101,1011\}]) \\
& \vee\; ([x_1x_2 \in \{01\}] \quad\;\; \wedge [x_3x_4x_5x_6 \in \{0001,1111\}]) \\
& \vee\; ([x_1x_2 \in \{00,01\}] \wedge [x_3x_4x_5x_6 \in \{0011,0111,1101\}]) \\
& \vee\; ([x_1x_2 \in \{10\}] \quad\;\; \wedge [x_3x_4x_5x_6 \in \{1001,1111\}]) \\
& \vee\; ([x_1x_2 \in \{11\}] \quad\;\; \wedge [x_3x_4x_5x_6 \in \{1101\}]) \\
& \vee\; ([x_1x_2 \in \{10,11\}] \wedge [x_3x_4x_5x_6 \in \{0101,1011\}])
\end{aligned} \qquad (37)$$

（第一行对应组1，类型$\begin{smallmatrix}1\\0\end{smallmatrix}$，然后是组1，类型$\begin{smallmatrix}0\\1\end{smallmatrix}$，等等；最后一行对应组2和类型$\begin{smallmatrix}1\\1\end{smallmatrix}$）像$[x_3x_4x_5x_6 \in \{0010, 0101, 1011\}]$这样的函数是$\{x_3, x_4, x_5, x_6\}$的3个极小项的或。

通常我们能将真值表看成一个$2^k \times 2^{n-k}$的数组，其中有l组，每组各有$\lfloor 2^k/l \rfloor$或$\lceil 2^k/l \rceil$行。一个大小为m的组会有2^m个基本类型的列。我们对每个组i和每个非零类型t，构造一个合取$(g_{it}(x_1, \cdots, x_k)) \wedge (h_{it}(x_{k+1}, \cdots, x_n))$，其中对有$t$为1的组的行，$g_{it}$是$\{x_1, \cdots, x_k\}$的所有极小项的或；而对有类型$t$的组$i$的列，$h_{it}$是$\{x_{k+1}, \cdots, x_n\}$的所有极小项的或。所有这些合取$(g_{it} \wedge h_{it})$的或给出了$f(x_1, \cdots, x_n)$。

一旦我们选择了参数k和l，$1 \leqslant k \leqslant n-2$且$1 \leqslant l \leqslant 2^k$，计算过程以在$t(k) + t(n-k)$步内计算$\{x_1, \cdots, x_k\}$的所有极小项和$\{x_{k+1}, \cdots, x_n\}$的所有极小项开始（参见（30））。然后，对$1 \leqslant i \leqslant l$，我们令组$i$由$(x_1, \cdots, x_k)$的值的行构成，从而$(i-1)2^k/l \leqslant (x_1, \cdots, x_k)_2 < i2^k/l$；它包含$m_i = \lceil i2^k/l \rceil - \lceil (i-1)2^k/l \rceil$行。我们对$t \in S_i$构造所有函数$g_{it}$，那些行的$2^{m_i}-1$个非空子集的族；$2^{m_i}-m_i-1$个之前计算的极小项的或将完成任务。我们同时构造所有函数h_{it}，代表非零类型t的列；为了这个目的我们需要每个组i中至多有2^{n-k}个或运算，因为我们能把合适的类型t的每个极小项或到函数h中。最后我们计算$f = \bigvee_{i=1}^{l} \bigvee_{t \in S_i} (g_{it} \wedge h_{it})$；每个与运算由一个非必须的第一个或补偿到$h_{it}$。所以总代价最多是

$$t(k)+t(n-k)+(l-1)+\sum_{i=1}^{l}((2^{m_i}-m_i-1)+2^{n-k}+(2^{m_i}-2)) \tag{38}$$

我们要选择k和l使得这个上界最小化。习题52讨论了n较小时的最佳选择。而当n较大时，一个好的选择产生可证明的较优链，至少对大多数函数来说。

定理L 令$C(n)$表示最贵的n变量布尔函数的代价。那么当$n\to\infty$时，有

$$C(n)\geqslant\frac{2^n}{n}\left(1+\frac{\lg n}{n}+O\left(\frac{1}{n}\right)\right) \tag{39}$$

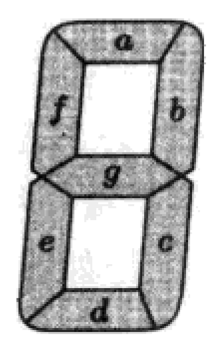
$$C(n)\leqslant\frac{2^n}{n}\left(1+3\frac{\lg n}{n}+O\left(\frac{1}{n}\right)\right) \tag{40}$$

证明：习题48表明下界（39）是定理S的结果。对于上界，我们在Lupanov的方法中设$k=\lfloor 2\lg n\rfloor$且$l=\lceil 2^k/(n-3\lg n)\rceil$；参见习题53。▮

合成优质链 公式（37）不是实现6位素数探测器的最好方法，但它提供了一个不错的策略。例如，我们不需要令变量x_1和x_2来决定行：习题51表明如果行是基于x_5x_6而列来自$x_1x_2x_3x_4$，会产生一条更好的链，而通常有很多途径通过摆弄k个变量和其他$n-k$个变量来分割真值表。

此外，我们能在（37）上通过使用我们关于所有4变量函数的完整知识进行改进；没有必要通过首先计算$\{x_3, x_4, x_5, x_6\}$的极小项来计算像$[x_3x_4x_5x_6\in\{0010, 0101, 1011\}]$这样的函数，如果我们知道从头开始计算每个这样的函数的最好方法。另一方面，我们确实需要同时计算一些4变量函数，所以极小项方法毕竟也许不是这么坏的想法。我们真的能改进它吗？

让我们尝试找到一个好方法来合成计算给定的4变量函数的集合的布尔链。（37）中$x_3x_4x_5x_6$的6个函数相当乏味（参见习题54），所以我们将考虑一个从日常生活中选取的更加有趣的例子。



七段显示器（seven-segment display）是一种如今无处不在的，以7块巧妙摆放的片段或明或暗来表示4位数字$(x_1x_2x_3x_4)_2$的方法。这些片段通常叫做(a, b, c, d, e, f, g)，如图所示。我们点亮片段(a, b, c, d, e, f)得到‘0’，而‘1’只用到了片段(b, c)。（顺便提一句，这种显示器的想法是由F. W. Wood在《U. S. Patent》974943 (1910) 中提出的，虽然Wood的最初设计用了8段，因为他认为‘4’需要1条对角线。）七段显示器通常只支持十进制数码‘0’，‘1’，…，‘9’；但当然计算机科学家的数码手表应该也能显示十六进制数码。所以我们会设计显示16个数码的七段逻辑

0123456789AbcdEF (41)

当给定各自的输入$x_1x_2x_3x_4=0000, 0001, 0010, \cdots, 1111$。

换句话说，我们要计算7个布尔函数，它们的真值表分别是

$$\begin{aligned}
a &= 1011\ 0111\ 1110\ 0011\\
b &= 1111\ 1001\ 1110\ 0100\\
c &= 1101\ 1111\ 1111\ 0100\\
d &= 1011\ 0110\ 1101\ 1110\\
e &= 1010\ 0010\ 1011\ 1111\\
f &= 1000\ 1111\ 1111\ 0011\\
g &= 0011\ 1110\ 1111\ 1111
\end{aligned} \tag{42}$$

如果我们只是想要分开计算每个函数，我们已经讨论过的一些方法会告诉我们怎样做，有代价$C(a)=5$，$C(b)=C(c)=C(d)=6$，$C(e)=C(f)=5$，以及$C(g)=4$；那么所有7个函数的总代价为37。但我们要找到一个单独的布尔链，包含全部的它们，而最短的这种链可以推测要高效很多。我们如何发现它呢？

好吧，找到一个对$\{a,b,c,d,e,f,g\}$确实最优化的链的任务从计算的立场来看大概是不可行的。但在之前解释的“足迹”想法的帮助下能找到一个惊人的好解答。即我们不仅知道如何计算一个函数的最小代价，而且知道如何计算所有在一条正规链中同那个最小代价相一致的第一步的集合。比如函数e具有代价5，但只有当我们以下列指令中的一条开始计算时

$$x_5=x_1\oplus x_4 \quad 或 \quad x_5=x_2\wedge \bar{x}_3 \quad 或 \quad x_5=x_2\vee x_3$$

幸运的是，这些富有吸引力的首步中的一个属于7个足迹中的4个：函数c、d、e、f和g都能以$x_5=x_2\oplus x_3$开始最优化的计算。所以那是个自然的选择；它本质上为我们节省了3步，因为我们知道原来的37步中最多需要完成33步。

现在我们能重新计算所有2^{16}个函数的代价和足迹，这和以前一样，但也初始化新函数x_5的代价到0。结果是函数c、d、f和g的代价减少了1，而足迹也改变了。例如，函数a仍有代价5，但当函数$x_5=x_2\oplus x_3$免费可用时，它的足迹已经从$\{x_1\oplus x_3, x_2\wedge x_3\}$增加到了$\{x_1\oplus x_3, x_1\wedge x_4, \bar{x}_1\wedge x_4, x_2\wedge x_3, \bar{x}_2\wedge x_4, x_2\oplus x_4, x_4\wedge x_5, x_4\oplus x_5\}$。

事实上，$x_6=\bar{x}_1\wedge x_4$对4个新足迹是共同的，所以我们又有了一条自然的途径继续下去。而当一切事物使用赋予了零代价的x_5和x_6重新计算后，后面的步骤$x_7=x_3\wedge \bar{x}_6$被证明在最新的5个足迹中是令人满意的。延续这种“贪婪”方式，我们不会总是如此幸运，但一条只有22步的非比寻常的链确实浮现在眼前。

$$\begin{array}{lll}
x_5=x_2\oplus x_3 & x_{13}=x_1\oplus x_7 & \bar{a}=x_{20}=x_{14}\wedge \bar{x}_{19}\\
x_6=\bar{x}_1\wedge x_4 & x_{14}=x_5\oplus x_6 & \bar{b}=x_{21}=x_7\oplus x_{12}\\
x_7=x_3\wedge \bar{x}_6 & x_{15}=x_7\vee x_{12} & \bar{c}=x_{22}=\bar{x}_8\wedge x_{15}\\
x_8=x_1\oplus x_2 & x_{16}=x_1\vee x_5 & \bar{d}=x_{23}=x_9\wedge \bar{x}_{13}\\
x_9=x_4\oplus x_5 & x_{17}=x_5\vee x_6 & \bar{e}=x_{24}=x_6\vee x_{18}\\
x_{10}=\bar{x}_7\wedge x_8 & x_{18}=x_9\wedge x_{10} & \bar{f}=x_{25}=\bar{x}_8\wedge x_{17}\\
x_{11}=x_9\oplus x_{10} & x_{19}=x_3\wedge x_9 & g=x_{26}=x_7\vee x_{16}\\
x_{12}=x_5\wedge x_{11} & &
\end{array} \tag{43}$$

（这是一条规范（normal）链，所以它包含范式化$\{\bar{a},\bar{b},\bar{c},\bar{d},\bar{e},\bar{f},g\}$而不是$\{a,b,c,d,e,f,g\}$。简单的改变能产生非规范函数而不改变代价。）

部分函数 实际中布尔函数的输出值往往只在特定的输入$x_1 \cdots x_n$下明确，而其他情况下的输出真的没有关系。比如我们也许知道，某些输入组合永远不会出现。这种情况下，我们在真值表的对应位置放一个星号，而非到处都是明确的0或1。

七段显示器提供了一个合适的案例，因为它的多数应用只包括10个二进制编码的十进制输入，其中我们有$(x_1x_2x_3x_4)_2 \leqslant 9$。我们不关心其他6种情况下哪段可见。所以（42）的真值表实际上变成了

$$
\begin{aligned}
a &= 1011\ 0111\ 11{*}{*}\ {*}{*}{*}{*}\\
b &= 1111\ 1001\ 11{*}{*}\ {*}{*}{*}{*}\\
c &= 1101\ 1111\ 11{*}{*}\ {*}{*}{*}{*}\\
d &= 1011\ 0110\ 11{*}{*}\ {*}{*}{*}{*}\\
e &= 1010\ 0010\ 10{*}{*}\ {*}{*}{*}{*}\\
f &= 1000\ 111{*}\ 11{*}{*}\ {*}{*}{*}{*}\\
g &= 0011\ 1110\ 11{*}{*}\ {*}{*}{*}{*}
\end{aligned}
\tag{44}
$$

（这里函数f在$x_1x_2x_3x_4 = 0111$处也有一个星号，因为‘7’显示为7或7都可以。在写这一节时这两种样式对作者来说在可用的显示单元出现的频率大约相当。截短的6和9的变体以前有时能见到，但谢天谢地它们已经消失了。）

真值表中的星号普遍被认为是无需关心的，这是一个只可能是电气工程师发明的古怪术语。表3表明任意选择输出的自由是有益的。例如，共有$\binom{16}{3}2^{13} =$ 4 587 520个真值表，其中3个无需关心；它们中的69%有代价4或更小，尽管如此只有21%的星号自由真值表允许这种节省。另一方面，无需关心节省掉的可能没有我们希望的那么多；习题63证明了一个在其真值表中有30%的无需关心的随机函数趋向于只能节省大约一个完全明确的函数的30%的代价。

计算（44）中7个部分明确的函数的最短的布尔链是什么？我们的贪婪足迹方法自身马上适应了无需关心的出现，因为我们能将符合某种有d个星号的模式的所有2^d个函数的足迹或在一起。分开计算每个函数的初始代价现在减少到$C(a) = 3$、$C(b) = C(c) = 2, C(d) = 5, C(e) = 2, C(f) = 3, C(g) = 4$，共只有21而非37。函数$g$并没有变便宜，但它却是有了个更大的足迹。像以前那样继续，但利用无需关心，我们现在能找到一条合适的长度仅为13的链——一条每个输出少于2个运算的链！：

$$
\begin{array}{lll}
x_5 = x_1 \oplus x_2 & \bar{e} = x_{10} = x_4 \vee x_8 & \bar{b} = x_{15} = x_2 \wedge \bar{x}_{13}\\
x_6 = x_3 \wedge \bar{x}_4 & g = x_{11} = x_7 \oplus x_8 & \bar{c} = x_{16} = \bar{x}_2 \wedge x_6\\
x_7 = x_1 \oplus x_3 & x_{12} = x_4 \oplus x_{11} & \bar{f} = x_{17} = \bar{x}_5 \wedge x_9\\
x_8 = x_2 \wedge \bar{x}_6 & \bar{d} = x_{13} = x_{10} \wedge x_{12} & \\
x_9 = x_3 \vee x_4 & \bar{a} = x_{14} = \bar{x}_3 \wedge x_{13} &
\end{array}
\tag{45}
$$

Tic-tac-toe 现在转到一个稍微大点的问题上，这是一个流行的儿童游戏。2个玩家轮流填充3×3格子的单元。一个玩家写×而另一个写○，持续下去直到有3个×或3个○出现在一条直线上（在那种情况下该玩家获胜）或所有9个单元填完而没有胜者（这种情况下是“cat’s game”或平手）。例如，游戏可能如此进行：

 (46)

表3 有d个无需关心的和代价为c的4变量函数的数目

	c=0	c=1	c=2	c=3	c=4	c=5	c=6	c=7
d=0	10	60	456	2 474	10 624	24 184	25 008	2 720
d=1	160	960	7 296	35 040	131 904	227 296	119 072	2 560
d=2	1 200	7 200	52 736	221 840	700 512	816 448	166 144	
d=3	5 600	33 600	228 992	831 232	2 045 952	1 381 952	60 192	
d=4	18 200	108 816	666 528	2 034 408	3 505 344	1 118 128	3 296	
d=5	43 680	257 472	1 367 776	3 351 488	3 491 648	433 568	32	
d=6	80 080	455 616	2 015 072	3 648 608	1 914 800	86 016		
d=7	114 400	606 944	2 115 648	2 474 688	533 568	12 032		
d=8	128 660	604 756	1 528 808	960 080	71 520	896		
d=9	114 080	440 960	707 488	197 632	4 160			
d=10	78 960	224 144	189 248	20 160				
d=11	41 440	72 064	25 472	800				
d=12	15 480	12 360	1 280					
d=13	3 680	800						
d=14	480							
d=15	32							
d=16	1							

这表示画×的一方赢了。我们的目标是设计一个最优化地玩tic-tac-toe的机器，当可以必胜时在胜点落子，当失败可以避免时不在败点落子。

更精确地，我们会把事情安排好，使得有18个布尔变量$x_1, \cdots, x_9, o_1, \cdots, o_9$控制当前位置单元的灯。单元编号为 ，就像电话键盘一样。单元j显示×，若$x_j = 1$；显示○若$o_j = 1$；或保持空白，若$x_j = o_j = 0$.*，我们不会有$x_j = o_j = 1$；因为那样会显示'⊗'。我们会假定变量$x_1, \cdots, x_9, o_1, \cdots, o_9$已经设置为指向一个合法的位置，此时没有人获胜；计算机用×，而现在轮到计算机走了。为了这个目的我们要定义9个函数$y_1, \cdots, y_9$，其中y_j意味着"将x_j从0变到1"。如果当前位置是平局，我们应该使$y_1 = \cdots = y_9 = 0$；否则恰有一个y_j应该等于1，而且当然只有当$x_j = o_j = 0$时输出值$y_j = 1$才会出现。

有了18个变量，我们的9个函数y_j都会有一个大小为$2^{18} = 262\,144$的真值表。结果只有4520个合法输入$x_1 \cdots x_9\ o_1 \cdots o_9$是可能的，所以那些真值表98.3%填充了无需关心的。如果我们希望设计和理解在直觉上有意义的一条布尔链，4520这个数目仍然大得令人不舒服。7.1.4节将讨论代表布尔函数的替代方法，用它能经常处理数百个变量，尽管相关的真值表大得不可思议。

该设置是基于20世纪50年代芝加哥科学和工业博物馆的一次展出，在那里作者首次得知开关电路的魔力。芝加哥的机器是贝尔电话实验室的W. Keister大约在1940年设计的，虽然让我先行，但我很快发现我是无法打败机器的。于是我决定尽可能很傻地移动，希望机器的设计得没有预料到有如此愚蠢的行为的存在。事实上，我使机器进入了一个位置，此时机器有两种取胜的移动，而它

同时都走了！走了两步，当然违反了规则，尽管机器宣告我输了，但我获得了精神上的胜利。

> 我着手考察一种称为"tit-tat-to"的游戏……
> 以查明所有可能的移动和位置的组合数。
> 我发现这相当重要。
> ……困难，然而，也随之而来。
> 当机器必须移子时，可能会出现两种不同的移动选择，
> 每个虽都一样导向赢局，
> 然而，除非给出规定，
> 机器可能会尝试两种相互矛盾的移动
> ——Charles Babbage，《Passages from the Life of a Philosopher》（1864）

多数18变量的函数要求多于$2^{18}/18$个门，但我们希望能做得更好。确实，一条貌似有理的策略马上毛遂自荐，以一些不难认定的条件的方式，在tic-tac-toe中走出合适的一着：

w_j，在单元j下×会获胜，完成×的一条线；

b_j，在单元j下○会失败，完成○的一条线；

f_j，在单元j下×会给予×方两种获胜方法；

d_j，在单元j下○会给予○方两种获胜方法。

例如，（46）中×需要下在中央来挡住○，所以属于b_5类型；幸运的是它也属于f_5类型，使得下一步必胜。

令$L = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{1,4,7\},\{2,5,8\},\{3,6,9\},\{1,5,9\},\{3,5,7\}\}$为获胜线路的集合。那么我们有

$$m_j = \bar{x}_j \wedge \bar{o}_j \text{[在单元}j\text{落子合法]} \quad (47)$$

$$w_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(x_i \wedge x_k) \text{[在单元}j\text{落子获胜]} \quad (48)$$

$$b_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(o_i \wedge o_k) \text{[在单元}j\text{落子阻挡]} \quad (49)$$

$$f_j = m_j \wedge S_2(\{\alpha_{ik} \mid \{i,j,k\} \in L\}) \text{[在单元}j\text{落子分叉]} \quad (50)$$

$$d_j = m_j \wedge S_2(\{\beta_{ik} \mid \{i,j,k\} \in L\}) \text{[在单元}j\text{落子防守]} \quad (51)$$

这里α_{ik}和β_{ik}表示一个单独的×或○以及一个空白，即

$$\alpha_{ik} = (x_i \wedge m_k) \vee (m_i \wedge x_k), \quad \beta_{ik} = (o_i \wedge m_k) \vee (m_i \wedge o_k) \quad (52)$$

例如，$b_1 = m_1 \wedge ((o_2 \wedge o_3 \vee (o_4 \wedge o_7) \vee (o_5 \wedge o_9))$；$f_2 = m_2 \wedge S_2(\alpha_{13}, \alpha_{58}) = m_2 \wedge \alpha_{13} \wedge \alpha_{58}$；$d_5 = m_5 \wedge S_2(\beta_{19}, \beta_{28}, \beta_{37}, \beta_{46})$。

有了这些定义，我们可以尝试对我们的落子排名。

$$\{w_1,\cdots,w_9\} > \{b_1,\cdots,b_9\} > \{f_1,\cdots,f_9\} > \{d_1,\cdots,d_9\} > \{m_1,\cdots,m_9\} \quad (53)$$

"能赢就赢；不能赢就挡，不能挡就分；不能分就防，不能防就合法地走一步。"此外，当选择合法落子时采用如下顺序似乎是明智的。

$$m_5 > m_1 > m_3 > m_9 > m_7 > m_2 > m_6 > m_8 > m_4 \qquad (54)$$

因为中间格5出现在4条获胜路线上，而走在角1、3、9或7上出现在3条中，边上的单元2、6、8或4只出现在2条中。我们也可以在（53）中的所有5组落子$\{w_j\}$、$\{b_j\}$、$\{f_j\}$、$\{d_j\}$和$\{m_j\}$中的下标采用该顺序。

为了确保至多选择一种落子，我们定义w'_j，b'_j，f'_j，d'_j，m'_j表示“之前的一个选择更好”。因此，$w'_5 = 0$，$w'_1 = w_5$，$w'_3 = w_1 \vee w'_1$，…，$w'_4 = w_8 \vee w'_8$，$b'_5 = w_4 \vee w'_4$，$b'_1 = b_5 \vee b'_5$，…，$m'_4 = m_8 \vee m'_8$。然后我们能通过令

$$y_j = (w_j \wedge \bar{w}'_j) \vee (b_j \wedge \bar{b}'_j) \vee (f_j \wedge \bar{f}'_j) \vee (d_j \wedge \bar{d}'_j) \vee (m_j \wedge \bar{m}'_j),\ \text{对于} 1 \leqslant j \leqslant 9 \qquad (55)$$

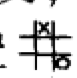
完成tic-tac-toe自动机器的定义。所以我们为m构造了9个门，为w构造了48个门，为b构造了48个门，为α和β构造了144个门，为f构造了35个门（在图9的帮助下），为d构造了35个门，为原始的变量构造了43个门，为y构造了80个门。此外我们能利用我们的4变量部分函数的知识将（52）中的6个运算减少到4个。

$$\alpha_{ik} = (x_i \oplus x_k) \vee \overline{(o_i \oplus o_k)},\quad \beta_{ik} = \overline{(x_i \oplus x_k)} \vee (o_i \oplus o_k) \qquad (56)$$

这套技巧节省了48个门；所以我们的设计共消耗了396个门。

（47）～（56）中对tic-tac-toe的策略在多数情况下运转良好，但也有一些显眼的小毛病。例如，它在棋局

 (57)

中由于无知而失败了。第二步×落子是d_3，防守○的分叉，然而它实际上迫使○在对面的角上分叉！另一种失败产生在，例如，在位置  后，当落子m_5导向和局 ，而不是出现在（46）中的×方获胜。习题65进行了补救，得到了一个需要445个门的完全正确的布尔tic-tac-toe玩家。

***函数分解**　如果函数$f(x_1, \cdots, x_n)$能写作$g(x_1, \cdots, x_k, h(x_{k+1}, \cdots, x_n))$的形式，通常先计算$y = h(x_{k+1}, \cdots, x_n)$，再计算$g(x_1, \cdots, x_k, y)$是个好主意。Robert L. Ashenhurst于1952年开辟了函数分解的研究方向［参见《Annals Computation Lab. Harvard University》29(1957), 74—116］，他观察到有一种简单的方法识别f是否具有这种特殊性质：如果我们像（36）中那样在一个$2^k \times 2^{n-k}$的数组中写下f的真值表，行的每项设置为$x_1 \cdots x_k$而列的每项设置为$x_{k+1} \cdots x_n$，那么想要的子函数g和h存在，当且仅当数组的列具有至多两个不同的值。例如，函数$\langle x_1 x_2 \langle x_3 x_4 x_5 \rangle\rangle$的真值表是

0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	0	1	0	1	1	1
1	1	1	1	1	1	1	1

当用这种二维形式表达时，一种类型的列对应情况$h(x_{k+1}, \cdots, x_n) = 0$；其他对应$h(x_{k+1}, \cdots, x_n) = 1$。

一般而言，变量$X = \{x_1, \cdots, x_n\}$可以分割为两个不相交的子集$Y = \{y_1, \cdots, y_k\}$和$Z = \{z_1, \cdots, z_{n-k}\}$，而我们可能有$f(x) = g(y, h(z))$。我们能通过观察$2^k \times 2^{n-k}$的真值表

的列来测试一对(Y, Z)分解，其中的行对应y的值。但有2^n种这样分割X的方法；且它们全部是潜在的赢家，除了当$|Y| = 0$或$|Z| \leqslant 1$时的不重要的情况。我们如何避免检查如此巨大数量的可能性呢？

V. Y. -S. Shen，A. C. McKellar和P. Weiner发现了一种实用的方法［《IEEE Transactions》C-20 (1971), 304—309］，他们的方法通常只需要$O(n^2)$步来识别任何可能存在的潜在有用的分割(Y, Z)。基本思想很简单：假定$x_i \in Z$, $x_j \in Z$且$x_m \in Y$。定义8个二元向量δ_l对$l = (l_1 l_2 l_3)_2$，其中δ_l在组件(i, j, m)中分别有(l_1, l_2, l_3)，其他地方是0。考虑任何随机选取的向量$x = x_1 \cdots x_n$，并计算$f_l = f(x \oplus \delta_l)$对于$0 \leqslant l \leqslant 7$。然后4个对

$$\binom{f_0}{f_1} \quad \binom{f_2}{f_3} \quad \binom{f_4}{f_5} \quad \binom{f_6}{f_7} \tag{58}$$

会出现在$2^k \times 2^{n-k}$真值表的一个2×4子矩阵中。所以一个分解不可行，如果这些对是截然不同的，或者如果它们含有3个不同的值。

让我们称这些对是“好”的，如果它们全部相等，或者如果它们只有两个不同的值。否则它们是“不好”的。如果f有重要的随机行为，我们很快会找到坏对，如果我们用几个不同的随机选取的向量x做这样的试验的话，因为对$f_0 f_1 \cdots f_7$的256种可能性中只有88种对应好对的集合；连续10次找到好对的几率只有$\left(\frac{88}{256}\right)^{10} \approx 0.00002$。而我们确实发现了坏对，因为选择$x_m \in Y$是不可能的，我们可以得出结论

$$x_i \in Z \text{ 且} x_j \in Z \Rightarrow x_m \in Z \tag{59}$$

例如假定$n = 9$且f是真值表$11001001000011\cdots00101$由$\pi$的512个最重要的位以二进制表示组成的函数。（这是我们在上面（5）和（6）中学习对于n=4的“more-or-less随机函数”。）很快找到该π函数的坏对，在每种情况（i, j, m）对$m \neq i < j \neq m$。确实，在作者的试验中，252种情况中的170种被立即决定了；每种情况的随机x向量的平均数只有1.52；而只有一种情况在坏对出现之前需要8个x那么多。因此（59）对所有相关的（i, j, m）成立，且这个函数显然不可分解。事实上，习题73指出我们不需要做252次测试来确立这个π函数的不可分解性；只要它们中的$\binom{n}{2} = 36$个就足够了。

转向一个更少随机的函数，令$f(x_1, \cdots, x_9) = (\det X) \bmod 2$，其中

$$X = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix} \tag{60}$$

当$i = 1, j = 2$且$m = 3$时，这个函数不满足条件（59），因为那种情况下没有坏对。但当$\{i, j\} = \{1, 2\}$时对$4 \leqslant m \leqslant 9$它确实满足（59）。我们能通过方便的缩写‘12⇒456789’表示这种行为；对所有对$\{i, j\}$，完整的蕴涵集合是

12⇒456789	18⇒34569	27⇒34569	37⇒24568	48⇒12369	67⇒12358
13⇒456789	19⇒24568	28⇒134679	38⇒14567	49⇒12358	68⇒12347
14⇒235689	23⇒456789	29⇒14567	39⇒124578	56⇒123789	69⇒124578
15⇒36789	24⇒36789	34⇒25789	45⇒123789	57⇒12369	78⇒123456
16⇒25789	25⇒134679	35⇒14789	46⇒123789	58⇒134679	79⇒123456
17⇒235689	26⇒14789	36⇒124578	47⇒235689	59⇒12347	89⇒123456

（参见习题69。）当我们随机调查该函数时，坏对稍微有点难找：当坏对确实出现时，在作者的试验中需要x的平均数上升到大约3.6。当然有必要对测试进行限制，通过选择一个可承受的界限t，然后当t次连续尝试寻找坏对失败时放弃。选择t=10能找到上面列出的198个蕴涵，除了其中8个。

像（59）的蕴涵是Horn子句，我们从7.1.1节知道，从Horn子句作进一步推论是很容易的。确实，使用习题74中的方法可推出，在查看少于50种情况(i, j, m)之后，只有$|Z| > 1$的可能分割是不重要的（$Y = \varnothing$，$Z = \{x_1, \cdots, x_9\}$）。

类似的结果出现在当$f(x_1, \cdots, x_9) = [\text{per } X > 0]$时，其中per表示固定（permanent）函数。（在这种情况下，f告诉我们是否在$K_{3,3}$的偶子图中有一个配对，其中的边由变量$x_1 \cdots x_9$声明。）现在只有180个蕴涵。

12⇒456789	18⇒3459	27⇒3459	37⇒2468	48⇒1269	67⇒1358
13⇒456789	19⇒2468	28⇒134679	38⇒1567	49⇒1358	68⇒2347
14⇒235689	23⇒456789	29⇒1567	39⇒124578	56⇒123789	69⇒124578
15⇒3678	24⇒3678	34⇒2579	45⇒123789	57⇒1269	78⇒123456
16⇒2579	25⇒134679	35⇒1489	46⇒123789	58⇒134679	79⇒123456
17⇒235689	26⇒1489	36⇒124578	47⇒235689	59⇒2347	89⇒123456

其中只有122个会在t=10作为切断阈值时被发现（t的最佳选择还不明确；也许应该动态变化）。那122条Horn子句对确立不可分解性仍绰绰有余。

可分解函数如何呢？对$f = \langle x_2x_3x_6x_9\langle x_1x_4x_5x_7x_8\rangle\rangle$我们取$i \wedge j \Rightarrow m$(对所有$m \notin \{i, j\}$)，除了当$\{i, j\} \subseteq \{1, 4, 5, 7, 8\}$时；后一种情况，$m$也必须属于$\{1, 4, 5, 7, 8\}$。虽然这212个蕴涵中只有185个在承受边界$t = 10$时被发现，分割$Y = \{x_2, x_3, x_6, x_9\}$，$Z = \{x_1, x_4, x_5, x_7, x_8\}$作为一种强烈的可能性很快浮现。

每当一种潜在分解被证据支持，我们需要验证对应的$2^k \times 2^{n-k}$真值表确实只有1个或2个截然不同的列。但我们很高兴花费2^n单位的时间在验证上，因为我们已大大地简化了f的计算。

比较函数$f = [(x_1x_2x_3x_4)_2 \geqslant (x_5x_6x_7x_8)_2 + x_9]$是另一种有趣的情况。它的184个潜在可推知的蕴涵是

12⇒3456789	18⇒2345679	27⇒34689	37⇒489	48⇒9	67⇒23489
13⇒2456789	19⇒2345678	28⇒34679	38⇒479	49⇒8	68⇒23479
14⇒2356789	23⇒46789	29⇒34678	39⇒478	56⇒1234789	69⇒23478
15⇒2346789	24⇒36789	34⇒789	45⇒1236789	57⇒1234689	78⇒349
16⇒2345789	25⇒1346789	35⇒1246789	46⇒23789	58⇒1234679	79⇒348
17⇒2345689	26⇒34789	36⇒24789	47⇒389	59⇒1234678	89⇒4

而当$t = 10$时找到了它们中的145个。这种情况下3个分解揭示了它们本身，分别有

$Z=\{x_4,x_8,x_9\}$，$Z=\{x_3,x_4,x_7,x_8,x_9\}$和$Z=\{x_2,x_3,x_4,x_6,x_7,x_8,x_9\}$。Ashenhurst证明了只要找到一个重要的分解，我们就能立即约简f；当我们尝试约简更简单的函数g和h时，其他分解会稍后浮现。

***部分函数的分解** 当函数f只是部分声明时，其分拆（Y, Z）取决于能够对无需关心的部分进行赋值，使得至少有两个不同的列出现在对应的$2^k\times 2^{n-k}$真值表中。

两个向量$u_1\cdots u_m$和$v_1\cdots v_m$由0、1和*构成，称它们不兼容，对某些j要么$u_j=0$且$v_j=1$，要么$u_j=1$且$v_j=0$，等价于如果由u和v指定的m立方体的子立方体没有共同的点。考虑图的顶点是有无需关心的真值表的列，其中u—v当且仅当u和v不兼容。我们能对*赋值从而得到至多两个截然不同的列当且仅当这个图是偶图。若u_1, ⋯, u_l相互兼容，它们一般化的一致$u_1\sqcup\cdots\sqcup u_l$定义在7.1.1节的习题32中，与它们全部兼容。[参见S. L. Hight,《IEEE Trans》C-22 (1973), 103—110; E. Boros, V. Gurvich、P. L. Hammer、T. Ibaraki和A. Kogan,《Discrete Applied Math》62 (1995), 51—75。] 因为一个图是偶图当且仅当它不含有奇回路，我们能通过深度优先搜索轻易地测试这个条件（参见7.4.1节）。

因此Shen、McKellar和Weiner的方法当无需关心出现时也能工作：(58) 中的4对被认为是坏的，当且仅当它们中的3个互不兼容。我们几乎能像以前一样运算，虽然坏对当有大量*的时候自然更难找到（参见习题72）。尽管如此，Ashenhurst的定理不再适用了。当一些分解存在时，它们都必须进一步探索，因为它们可能采用了不同的无需关心设置，而其中一些可能比另一些更好。

尽管多数函数$f(x)$没有简单分解$g(y, h(z))$，我们不必太快放弃希望，因为其他的如$g(y, h_1(z), h_2(z))$的形式也可能导致一条有效的链。例如，若f在它的3个变量$\{z_1, z_2, z_3\}$上是对称的，我们总是能写出$f(x)=g(y, S_{1,2}(z_1,z_2,z_3), S_{1,3}(z_1,z_2,z_3))$，因为$S_{1,2}(z_1,z_2,z_3)$和$S_{1,3}(z_1,z_2,z_3)$赋予了值$z_1+z_2+z_3$。（注意，只要4步就足够计算$S_{1,2}$和$S_{1,3}$了。）

一般来说，像H. A. Curtis [《JACM》8 (1961), 484—496]观察到的那样，$f(x)$能表达为形式$g(y, h_1(z), \cdots, h_r(z))$当且仅当对应$Y$和$Z$的$2^k\times 2^{n-k}$真值表有至多$2^r$个不同的列。而当无需关心出现时，相同结果成立当且仅当Y和Z的不兼容性图能以至多2^r种颜色着色。

例如，上面考虑的函数$f(x)=(\det X) \bmod 2$被证明有8个截然不同的列，当$Z=\{x_4,x_5,x_6,x_7,x_8,x_9\}$；那是个令人惊讶的小数目，考虑到真值表有8行以及64列。如果我们还不知道这样一条规则，从这个事实我们也许会发现如何利用第一行的余子式展开行列式

$$f(x)=x_1\wedge h_1(x_4,\cdots,x_9)\oplus x_2\wedge h_2(x_4,\cdots,x_9)\oplus x_3\wedge h_3(x_4,\cdots,x_9)$$

当有$d\leqslant 2^r$个不同的列时，我们能把$f(x)$看成一个y和$h(z)$的函数，其中h取每个二元向量$z_1\cdots z_{n-k}$为$\{0, 1, \cdots, d-1\}$中的一个值。因此$(h_1, \cdots, h_r)$实质上是对不同列类型的编码，并且我们希望找到非常简单的函数$h_1, \cdots, h_r$来提供这样的编码。此外，如果d严格小于2^r，函数$g(y, h_1, \cdots, h_r)$会有许多无需关心，也会减少它的代价。

不同的列可能也暗示了一个函数g，其h中有无需关心。例如，我们能用$g(y_1, y_2, h_1, h_2) = (y_1 \oplus (h_1 \wedge y_2)) \wedge h_2$，当所有列不是$(0, 0, 0, 0)^T$就是$(0, 0, 1, 1)^T$或是$(0, 1, 1, 0)^T$时；那么$h_1(z)$的值是任意的，当$z$对应一个全零的列。H. A. Curtis阐明了如何发掘这个想法，当$|Y| = 1$且$|Z| = n-1$时[参见《IEEE Transactions》C-25 (1976), 1033—1044]。

对分解技术的广泛讨论，参见Richard M. Karp，《J. Society for Industrial and Applied Math》11 (1963), 291—335。

***n*的较大值** 我们仅仅已经考虑了相当微小的布尔函数的例子。定理S告诉我们较大的、随机的例子天生是困难的；但实际的例子可能也是高度非随机的。所以利用启发式方法搜索简化是有意义的。

当n增长时，目前已知的处理布尔函数最好的办法是从一条布尔链开始，而不是一张巨大的真值表，并且尝试通过"局部改动"去改进那条链。这条链可以用一个等式的集合声明。然后，如果一个中间结果被用在比较少的后续步骤中，我们能尝试排除它，暂时性地将那些后续步骤变成3变量函数，并且如有可能，重新以公式表示那些函数，来构造一条更好的链。

例如，假定门$x_i = x_j \circ x_k$只用了一次，在门$x_l = x_i \circ x_m$中，从而$x_l = (x_j \circ x_k) \Box x_m$。其他门可能已经存在，由我们计算其他$x_j$、$x_k$和$x_m$的函数得到；而$x_j$、$x_k$和$x_m$的定义也许意味着某些$(x_j, x_k, x_m)$的连接值是不可能的。因此我们也许能够通过只多作一步运算从其他门计算x_l。例如，若$x_i = x_j \wedge x_k$且$x_l = x_i \vee x_m$，并且若$x_j \vee x_m$和$x_k \vee x_m$的值出现在链的其他位置，我们能设$x_l = (x_j \vee x_m) \wedge (x_k \vee x_m)$；这排除了$x_i$并将代价减少了1。或者如果$x_j \wedge (x_k \oplus x_m)$出现在其他位置，我们知道$x_j x_k x_m \neq 101$，我们能设$x_l = x_m \oplus (x_j \wedge (x_k \oplus x_m))$。

如果x_i只用在x_l中且x_l只用在x_p中，那么门x_p依赖4个变量，而我们也许能通过使用我们关于4变量函数的全部知识减少代价，用一种更好的方法得到x_p而排除x_i和x_l。类似地，若x_i只出现在x_l和x_p中，我们能排除x_i，如果我们能找到一种更好的方法来计算两个不同的4变量函数，可能含有无需关心，也可能含有其他免费提供的4变量函数。通过采用上面讨论的足迹方法，我们又一次知道了如何解决这样的问题。

当没有局部改动能减少代价时，我们也能尝试维持甚至增加代价的局部改动，来发现不同种类的也许能用其他方法简化的链。我们会在7.10节中广泛讨论这样的局部搜索方法。

布尔优化技术的杰出综述，电气工程师称此问题为"多层次逻辑综合"，由R. K. Brayton, G. D. Hachtel和A. L. Sangiovanni-Vincentelli发表在《Proceedings of the IEEE》78 (1990), 264—300，以及G. De Micheli(McGraw-Hill, 1994)写的《Synthesis and Optimization of Digital Circuits》一书中。

下界 定理S告诉我们几乎每个有$n \geq 12$变量的布尔函数，由于需要一条长度超过$2^n/n$的链，因此都是很难计算的。然而从包括代表数千个布尔变量的电信号的逻辑电路建立起来的现代计算机，每微秒都愉快地计算着天文数字的布尔函数。

虽然存在定理S，但显然有大量重要函数能快速及时得到计算。确实，定理的证明是非直接的；我们简单计数低代价的情况，所以我们完全不知道任何实际中可能出现的特例。当我们想要计算一个给定的函数，而我们只能想到一种吃力的方法来做这项工作时，我们如何确保不存在技巧性的捷径呢？

这个问题的答案几乎是丢人的：经过数十年全力以赴的研究，计算机科学家无法找到任何明确的函数$f(x_1, \cdots, x_n)$家族，它们的代价随着n的增长是天然非线性的。真实的行为是$2^n/n$，但没有强如$n \log\log\log n$的下界已被证明！当然我们能拼凑人工实例，诸如“词典学上长度2^n的最小真值表不能由任何长度为$\lfloor 2^n/n \rfloor - 1$的布尔链得到”；但这样的函数无疑是不明确的。一个明确的函数$f(x_1, \cdots, x_n)$的真值表应当在比如最多2^{cn}个单位时间内可计算，对某个常数c；也就是说，声明所有函数值需要的时间应当是真值表长度的多项式。在那些基础规则下，目前被认为没有单输出函数家族具有联合复杂度超过$3n + O(1)$，随着$n \to \infty$。[参见N. Blum,《Theoretical Computer Science》28 (1984), 337—345。]

眼前的图景并不完全凄凉，因为对实践中的重要函数的一些有趣的线性下界已经被证明了。N. P. Red'kin在1970年介绍了一种得到这样的结果的基本方法：假定我们有一条$f(x_1, \cdots, x_n)$的代价为r的最优链。通过设$x_n \leftarrow 0$或$x_n \leftarrow 1$，我们得到函数$g(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 0)$和$h(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 1)$的约简了的链，若$x_n$用作$u$个不同门的输入，有代价$r-u$。此外，若$x_n$用在一个“渠限化（canalizing）”门$x_i = x_n \circ x_k$，其中运算符$\circ$既不是$\oplus$也不是$\equiv$，$x_n$的某些设置将迫使$x_i$为常量，从而进一步归约$g$或$h$的链。$g$和/或$h$上的下界因此导致了一个$f$上的下界。（参见习题77—81。）

但非线性下界的证明在哪？几乎每个具有是-非回答的问题都能表达为布尔函数，所以我们不知道怎样在线性时间，甚至多项式时间内估计的确切函数并不缺乏。例如，任何有顶点$\{v_1, \cdots, v_m\}$的有向图G能由它的邻接矩阵$\boldsymbol{X}$代表，其中$x_{ij} = [v_i \to v_j]$；那么

$$f(x_{12}, \cdots, x_{1m}, \cdots, x_{m1}, \cdots, x_{m(m-1)}) = [G\text{有一条哈密顿路径}] \qquad (61)$$

是一个$n = m(m-1)$变量的布尔函数。我们会非常高兴能在比如说n^4步内计算这个函数。我们确实知道如何在$O(m!2^n) = 2^{n+O(\sqrt{n}\log n)}$步内计算$f$的真值表，因为只存在$m!$条潜在的哈密顿路径；因此$f$确实是“确切”的。但没有人知道如何在多项式时间内计算f，或者如何证明没有一条$4n$步的链。

尽管我们知道，对每个n、f的较短的布尔链可能存在。毕竟，图9和图10揭示了非常灵巧的链的存在，甚至在4变量和5变量的情况下。对我们将需要解决的所有更大的问题的高效的链可能也“就在那里”——然而完全在我们的掌握之外，因为我们没有时间找到它们。即使某个无所不知的存在向我们揭示了简单的链，我们也会觉得它们不可思议，因为对它们的正确性的最短证明的长度值也比我们大脑的细胞数要多。

对于大部分布尔函数，定理S排除了此种场景。但在整个世界历史长河中，有着实际重要性的布尔函数的数目将接近2^{100}个，定理S却没有我们关于它们的任何

事情。

尽管如此，在1974年，Larry Stockmeyer和Albert Meyer已经可以构造布尔函数f了，其复杂度是可证巨大的。但他们的f并非“显式的”，以上面讨论的含义来说，也不是人工的；它在数学逻辑中自然出现。考虑像这样的符号化声明

`048+1015≠1063` (62)

`∀m∃n(m<n+1)` (63)

`∀n∃m(m+1<n)` (64)

`∀a∀b(b≥a+2⇒∃ab(a<ab∧ab<b))` (65)

`∀A∀B(A≡B⇔¬∃n(n∈A∧n∉B∨n∈B∧n∉A))` (66)

`∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≤n)))` (67)

`∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≥n)))` (68)

`∃P∀a((a∈P⇔a+3∉P)⇔a<1000)` (69)

`∀A∀B(∀C∀c(C≡A∧c=1∨C≡B∧c=0⇒(∀n(n∈C⇔n+1∈C)⇔c=1))⇒¬A≡B)` (70)

Stockmeyer和Meyer通过采用如下63个符号的字母表定义语言L，

`∀∃¬()≡∈∉+∧∨⇒⇔<≤=≠≥>abcdefghijklmnopqABCDEFGHIJKLMNOPQ0123456789`

并赋予这些符号以常规意义。L语句中的小写字母字符串，如（65）中的‘*ab*’代表数值变量，限制在非负整数；大写字母字符串代表集合变量，限制在这些数的有限集合。例如，（66）意味着“对所有有限集合A和B，我们有$A = B$当且仅当不存在一个数n在A中但不在B中，或在B中但不在A中。”这些声明中的一些为真，另一些为假。（参见习题82。）

字符串（62）～（70）全部属于L，但该语言实际上相当受限：一个数上允许的唯一算术运算是加上一个常数；我们可以写‘`a+13`’但不能写‘`a+b`’。数和集合之间允许的唯一关系是属于（∈或∉）。集合之间允许的唯一关系是相等（≡）。此外所有变量必须由∃或∀量化[⊖]。

L的每条长度$k \leqslant n$的语句能由一个长度为$6n$的二元向量代表，其中最后$6(n-k)$位是零。令$f(x)$为一个$6n$变量的布尔函数，使得$f(x) = 1$，只要x代表L中的一条为真的语句；$f(x) = 0$，只要x代表一条为假的语句；$f(x)$的值为不确定，当x不代表一条有意义的语句。这样的函数f的真值表能在有限步数内构造，根据Büchi和Elgot的定理［《Zeitschrift für math. Logik und Grundlagen der Math》6 (1960), 66—92;《Transactions of the Amer. Math. Soc》98 (1961), 21—51］。但“有限”并不意味着“可行”：Stockmeyer和Meyer证明了

$$C(f)>2^{r-5} \quad 只要 n \geqslant 460+0.302r+5.08\ln r 且 r>36 \tag{71}$$

特别地，当$n = 621$时，我们有$C(f) > 2^{426} > 10^{128}$。一条有那么多门的布尔链永远无法建立，因为$10^{128}$是宇宙中质子数目的一个宽厚的上界。所以这是一个永远无法解决的相当小而有限的问题。

⊖ 从技术上说，L中的语句属于“弱二阶带一个后继的一元逻辑”（weak second-order monadic logic with one successor），弱二阶逻辑允许在有穷集上使用量词，带k个后继的一元逻辑是关于无标记k叉树的理论。

Stockmeyer和Meyer的证明细节见《JACM》49 (2002), 753—784。基本思想是语言L尽管有限，但足够丰富，使用相当短的语句就足以描述真值表和布尔链的复杂度；因此f不得不处理本质上指向它自己的输入。

***进一步阅读** 关于布尔门网络已经发表有数千篇重要的论文，该类网络是多个领域的理论和实践基础。本节集中在与计算机编程和对于相关的主题上。但其他主要和并行计算相关的主题也得到了广泛的研究，如浅深度电路的研究，其中的门可有任何数目的输入（“无限制扇入”）。Ingo Wegener的著作《The Complexity of Boolean Functions》(Teubner和Wiley, 1987)给出了整个主题较好的介绍。

我们主要考虑所有二元运算符有相同重要性的布尔链。为此，像$\oplus$或$\overline{\subset}$这样的门多少都跟$\wedge$或$\vee$这样的门一样有吸引力。但当计算单调函数时，想知道是否只使用单调运算符$\wedge$和$\vee$也可行是自然的。Alexander Razborov给出的引人注目的证明技术表明，事实上，单调运算符本身天然具有有限的能力。例如，他证明了，所有确定一个$n \times n$的0和1的矩阵是否为零或非零的AND-OR链必然有代价$n^{\Omega(\log n)}$。[参见《Doklady Akademii Nauk SSSR》281 (1985), 798—801；《Matematicheskie Zametki》37 (1985), 887—900。]相反，我们在7.5.1节会看到这个问题，等价于“偶匹配”，在仅仅$O(n^{2.5})$步内是可解决的。此外，当我们在$\wedge$和$\vee$之外允许取反或其他布尔操作外，那一节中的高效方法能实现为仅仅有稍大一点代价的布尔链。(Vaughan Pratt称此为“反面思考的力量”。) 对Razborov的方法的介绍出现在习题85和86中。

习　题

1. [24] 式（6）中的“随机”函数涉及一个代价为4、深度为4的布尔链。请找到一个具有同样代价的深度为3的式子。

2. [21] 请指出如何利用深度为3、代价为5的式子来计算（a）$w\oplus\langle xyz\rangle$ 和 (b) $w\wedge\langle xyz\rangle$ 。

3. [M23] (B. I. Finikov, 1957) 如果布尔函数$f(x_1,\cdots,x_n)$ 精确地在k个点处为真，请证明$L(f) < 2n+(k-2)2^{k-1}$。提示：考虑 k=3 和n=10^6。

4. [M26] (P. M. Spira,1971) 请证明一个布尔函数的最小深度和式子长度满足$\lg L(f) < D(f) \leqslant \alpha \lg L(f)+1$，其中$\alpha=2/\lg\left(\frac{3}{2}\right)\approx 3.419$。提示：每个拥有$r\geqslant 3$个内部节点的二叉树包含一个拥有$s$个内部节点的子树，其中 $\frac{1}{3}r\leqslant s<\frac{2}{3}r$。

▶ 5. [21] Fibonacci阈函数$F_n(x_1,\cdots,x_n)=\langle x_1^{F_1}x_2^{F_2}\cdots x_{n-1}^{F_{n-1}}x_{n-2}^{F_{n-2}}\rangle$在7.1.1节的习题101中被分析过，其中$n\geqslant 3$。是否存在一种有效的方法来计算它呢？

6. [20]判断正误：布尔函数$f(x_1,\cdots,x_n)$是规范的，当且仅当它满足分配律$f(x_1,\cdots,x_n)\wedge y=f(x_1\wedge y,\cdots,x_n\wedge y)$。

7. [20] 将布尔链‘$x_5=x_1\overline{\vee}x_4$, $x_6=\bar{x}_2\vee x_5$, $x_7=\bar{x}_1\wedge\bar{x}_3$, $x_8=x_6=x_7$’转换成一个等价的链（$\hat{x}_5$, $\hat{x}_6$, $\hat{x}_7$, $\hat{x}_8$），并使得每一步都是规范的。

▶ 8. [20] 解释为什么(11)是变量x_k的真值表。

9. [20] 算法L决定了所有函数f的最短表达式的长度，但是它没有提供进一步的信息。请扩展这种算法，使得它也能像(6)一样提供精确的最短长度的式子。

▶ 10. [20] 修改算法L，使得它计算$D(f)$而不是$L(f)$。

▶ 11. [22] 修改算法L，使得它不是计算长度$L(f)$，而是像正文中介绍的那样，计算上界$U(f)$和足迹$\phi(f)$。

12. [15] 什么样的布尔链与最小存储模式(13)等价？

13. [16] 例(13)中f_1、f_2、f_3、f_4和f_5的真值表是什么？

14. [22] 当给出g的真值表的条件下，怎样以一种简单的方法来计算(17)的$5n(n-1)$真值表？（使用如同(15)和（16）的按位操作。）

15. [28] 请使用最小的存储空间找到尽可能短的途径来计算下列函数：(a) $S_2(x_1, x_2, x_3, x_4)$; (b) $S_1(x_1, x_2, x_3, x_4)$;(c)(18)中的函数。

16. [HM33]证明2^{128}个布尔函数$f(x_1,\cdots, x_7)$中有少于2^{118}个能够以最小的存储空间来计算。

▶ 17. [25](M.S.Paterson, 1977)尽管布尔函数$f(x_1,\cdots,x_n)$不能在任何情况下用n个寄存器来计算，请证明$n+1$个寄存器在任何情况下都是足够的。换句话说，请证明在任何情况下都存在像(13)一样的操作序列来计算$f(x_1,\cdots x_n)$，如果我们规定$0\leqslant j(i)$, $k(i)\leqslant n$。

▶ 18. [35]研究f $(x_1, x_2, x_3, x_4, x_5)$的优化最小存储空间计算，并指出多少类的5变量函数有$C_m(f)=r$,对于$r=0$，1，2，…？

19. [M22]如果一个布尔链使用n个变量，并且长度$r<n+2$，请证明它要么是从顶到底的构造，要么是从底到顶的构造。

▶ 20. [40](R. Schroeppel, 2004) 一个布尔链若不使用操作符⊕或者≡，那么就称它为渠限化的。在这个条件下找出所有4变量函数的最优代价、长度和深度。足迹启发式方法仍然给出最优化的结果吗？

21. [46]1951年哈佛研究者发现了多少个4变量函数的最优电子管线路？

22. [21]解释图10中对于S_3的链，注意它合并了图9中对于$S_{2,3}$的链。请为$S_2(x_1, x_2, x_3, x_4, x_5)$找出相似的链。

▶ 23. [23]图10仅仅描述了基于5个元素的64个对称函数中的16个。请解释怎样为其他的函数写最优链。

24. [47]每个对称函数f都有$C_m(f)=C(f)$吗？

▶ 25. [17]假设我们想要一个包含所有n变量函数的布尔链：设$f_k(x_1,\cdots,x_n)$为一个函数，它的真值表是k的二进制表示，其中$0\leqslant k<m=2^{2^n}$，那么$C(f_0f_1\cdots f_{m-1})$是多少？

26. [25]判断正误：如果$f(x_0,\cdots,x_n)=(x_0\wedge g(x_1,\cdots,x_n))\oplus h(x_1,\cdots,x_n)$，其中$g$和$h$是非平凡布尔函数，它们的连接代价是$C(gh)$,那么$C(f)=2+C(gh)$。

▶ 27. [23]一个全加器（22）能够以最小的存储空间在5步以内实现吗（也就是说，完全以3个1bit寄存器）？

28. [26]证明$C(u'v')=C(u''v'')=5$,其中两输出函数定义为$(u'v')_2=(x+y-(uv)_2)$ mod

4, $(u''v'')_2=(-x-y-(uv)_2) \bmod 4$。使用这些函数以少于$2.5n$步来计算$[(x_1+\cdots+x_n) \bmod 4=0]$。

29. [M28]证明正文中对于旁路加法（27）的路径有深度$O(\log n)$。

30. [M25]为旁路加法的代价$s(n)$解决二进制循环(28)。

31. [21]如果$f(x_1,\cdots,x_n)$对称，请证明$C(f)\leqslant 5n+O(n/\log n)$。

32. [HM16]为什么（30）的解答满足$t(n)=2^n+O(2^{n/2})$?

33. [HM22]判断正误：如果$1\leqslant N\leqslant 2^n$，那么$\{x_1,\cdots,x_n\}$的前$N$个最小项都能够在$N+O(\sqrt{N})$步内计算出来，其中$n\to\infty$且$N\to\infty$。

▸ 34. [22]一个优先级编码器有$n=2^m-1$个输入$x_1\cdots x_n$和m个输出$y_1\cdots y_m$，其中$(y_1\cdots y_m)_2=k$当且仅当$k=\max\{j|j=0或者x_j=1\}$。设计一个有代价为$O(n)$和深度为$O(m)$的优先级编码器。

35. [23]如果$n>1$,请证明合取$x_1\wedge\cdots\wedge x_{k-1}\wedge x_{k+1}\wedge\cdots\wedge x_n$（其中$1\leqslant k\leqslant n$）都能够以总代价$\leqslant 3n-6$由$(x_1,x_2,\cdots,x_n)$计算出来。

▸ 36. [M28](R. E. Ladner和M. J. Fischer,1980)设y_k是“前缀”$x_1\wedge\cdots\wedge x_k$，其中$1\leqslant k\leqslant n$。很明显$C(y_1\cdots y_n)=n-1$且$D(y_1\cdots y_n)=\lceil\lg n\rceil$；但我们不能同时最小化代价和深度。请找出一个最优深度$\lceil\lg n\rceil$的链，其代价$<4n$。

37. [M28](Marc Snir, 1986)设$n\geqslant m\geqslant 1$,考虑以下的算法：

S1. [向上循环。] 对于$t\leftarrow 1, 2,\cdots, \lceil\lg m\rceil$,设$x_{\min(m,2^t k)}\leftarrow x_{2^t(k-1/2)}\wedge x_{\min(m,\,2^t k)}$，其中$k\geqslant 1$且$2^t(k-1/2)<m$。

S2. [向下循环。] 对于$t\leftarrow\lceil\lg m\rceil-1, \lceil\lg m\rceil-2,\cdots,1$,设$x_{2^t(k+1/2)}\leftarrow x_{2^t k}\wedge x_{2^t(k+1/2)}$,其中$k\geqslant 1$且$2^t(k+1/2)<m$。

S3. [扩展。] 对于$k\leftarrow m+1, m+2, \cdots, n$, 设$x_k\leftarrow x_{k-1}\wedge x_k$。▮

a) 请证明这个算法解决了习题36的前缀问题：它将$(x_1,x_2,\cdots,x_n)$转化为$(x_1,x_1\wedge x_2,\cdots,x_1\wedge x_2\wedge\cdots\wedge x_n)$。

b) 设$c(m,n)$和$d(m,n)$分别是相关的布尔链的代价和深度。请证明如果n足够大，那么$c(m,n)+d(m,n)=2n-2$。

c) 假设n的值确定，那么$d(n)=\min_{1\leqslant m\leqslant n} d(m,n)$的值是多少？请证明$d(n)<2\lg n$。

d) 证明对于前缀问题存在一个代价为$2n-2-d$和深度为d的布尔链，只要是当$d(n)\leqslant d<n$时。（这个代价是最优的，参见习题81。）

38. [25]在5.3.4节中我们学习过排序网络，通过其中$\hat{S}(n)$个比较器模块能够将n个数字$(x_1,x_2,\cdots,x_n)$按升序排序。如果输入x_j是0和1，每个比较器模块等价于两个门$(x\wedge y,\ x\vee y)$;因此一个排序网络与一个特定类型的布尔链相关，这种布尔链计算n个特定函数$(x_1, x_2,\cdots,x_n)$。

a) 排序网络计算的n个函数$f_1f_2\cdots f_n$分别是多少？

b) 请证明那些函数$\{f_1, f_2,\cdots, f_n\}$能够在$O(n)$步内以一个深度为$O(\log n)$的链计算出来。(因此排序网络不是渐近最优的。)

▸ 39. [M21] (M. S. Paterson和P. Klein, 1980)实现(31)中2^m路多路复用器$M_m(x_1\cdots,x_m;\ y_0,\ y_1,\cdots,y_{2^m-1})$，方法是通过使用一个有效的链来同时建立上界$C(M_m)\leqslant$

$2n+O(\sqrt{n})$和$D(M_m)\leqslant m+O(\log m)$。

40. [25]如果$n\geqslant k\geqslant 1$，设$f_{nk}(x_1,\cdots,x_n)$是"连续k个"的函数，

$$(x_1\wedge\cdots\wedge x_k)\vee(x_2\wedge\cdots\wedge x_{k+1})\vee\cdots\vee(x_{n+1-k}\wedge\cdots\wedge x_n)$$

请证明这个函数的代价$C(f_{nk})$小于$4n-3k$。

41. [M23]（有条件之和的加法器）一种以深度$O(\log n)$实现二进制加法(25)的方法是基于习题4的多路复用器策略：如果$(xx')_2+(yy')_2=(zz')_2$，其中$|x'|=|y'|=|z'|$，那么可得$(x)_2+(y)_2=(z)_2$和$(x')_2+(y')_2=(z')_2$，或者$(x)_2+(y)_2+1=(z)_2$和$(x')_2+(y')_2=(1z')_2$。为了节省时间，我们能在计算$(x')_2+(y')_2$的同时计算$(x)_2+(y)_2$和$(x)_2+(y)_2+1$。然后，当我们知道这个不太重要的部分$(x')_2+(y')_2$是否产生一个进位时，我们就能够使用多路复用器为最重要的部分选择正确的位。

如果这个方法被递归地用作由n位加法器来建立$2n$位加法器，那么当$n=2^m$时需要多少个门？相应的深度是多少？

42. [25]在二进制加法(25)中，设$u_k=x_k\wedge y_k$和$v_k=x_k\oplus y_k$，其中$0\leqslant k<n$。

a) 证明$z_k=v_k\oplus c_k$，其中进位c_k满足$c_k=u_{k-1}\vee(v_{k-1}\wedge(u_{k-2}\vee(v_{k-2}\wedge(\cdots(u_1\wedge v_0)\cdots))))$。

b) 设$U_k^k=0$, $V_k^k=1$，且$U_j^{k+1}=u_k\vee(v_k\wedge U_j^k)$, $V_j^{k+1}=v_k\wedge V_j^k$，其中$k\geqslant j$。请证明$c_k=U_0^k$，且$U_i^k=U_j^k\vee(V_j^k\wedge U_i^j)$，$V_i^k=V_j^k\wedge V_i^j$，其中$i\leqslant j\leqslant k$。

c) 设$h(m)=2^{m(m-1)/2}$。请证明当$n=h(m)$时，进位$c_1,\cdots,c_n$都能以深度$(m+1)m/2\approx\lg n+\sqrt{2\lg n}$和总代价$O(2^m n)$计算出来。

▶43. [28] 有限状态变换器是一个抽象机，拥有一个有限的输入字母表A、一个有限的输出字母表B和一个内部状态Q的有限集。那些状态中的一个状态q_0被称为"初始状态"。给定一个字符串$\alpha=a_1\cdots a_n$，其中每个$a_j\in A$，这个抽象机计算一个字符串$\beta=b_1\cdots b_n$，其中每个$b_j\in B$，如下所示。

T1. [初始化。] 设$j\leftarrow 1$且$q\leftarrow q_0$。

T2. [已完成？] 如果$j>n$，那么终止这个算法。

T3. [输出b_j。] 设$b_j\leftarrow c(q,a_j)$。

T4. [增加j的值。] 设$q\leftarrow d(q,a_j)$，$j\leftarrow j+1$，并返回到步骤T_2。▌

这个抽象机有内置的说明，规定对于每个状态$q\in Q$和每个字符$a\in A$有$c(q,a)\in B$和$d(q,a)\in Q$。这道习题的目的是为了表明，如果任意有限状态转换器的字母表A和B以二进制的形式编码，那么字符串β能够从α通过一个大小为$O(n)$和深度为$O(\log n)$的布尔链计算出来。

a) 考虑这个问题，它通过设置$b_j\leftarrow a_j\oplus[a_j=a_{j-1}=\cdots=a_{j-k}=1$且$a_{j-k-1}=0$，其中$k$是奇数]，将一个二进制向量$a_1\cdots a_n$变为$b_1\cdots b_n$，其中假定$a_0=0$。例如，$\alpha=1100100100011111101101010\mapsto\beta=1000100100010101001001010$。请证明这个转换能够通过一个有限状态转换器来执行，其中$|A|=|B|=|Q|=2$。

b) 假设一个有限状态转换器在读取$a_1\cdots a_{j-1}$后处于状态q_j。请解释如何使用习题36中的Ladner和Fischer构造来计算序列$q_1\cdots q_n$，使得它的布尔链的代价是$O(n)$、深度是$O(\log n)$。（从序列$q_1\cdots q_n$计算$b_1\cdots b_n$是一件容易的事情，因为$b_j=c(q_j,a_j)$。）

c) 将(b)中的方法应用到(a)中的问题。

▶44. [26] (R. E. Ladner和M. J. Fischer,1980) 请证明二进制加法(25)的问题可以看作是一个有限状态转换的问题。请描述当$n=2^m$时习题43中构造产生的布尔链，并将其与习题41中的有条件之和的加法器进行比较。

45. [HM20]为什么定理S的证明过程不简单地认为：选择$j(i)$和$k(i)$使得$1 \leqslant j(i), k(i)<i$的方法数是$n^2(n+1)^2\cdots(n+r-1)^2$？

▶46. [HM21]设$\alpha(n)=c(n, \lfloor 2^n/2 \rfloor)/2^{2^n}$是$n$变量布尔函数$f(x_1,\cdots,x_n)$的一部分，其中$C(f)\leqslant 2^n/n$。请证明当$n\to\infty$时，即有$\alpha(n)\to 0$。

47. [M23]将定理S扩展到n个输入和m个输出的函数。

48. [HM23]请找出最小的整数$r=r(n)$使得$(r-1)!2^{2^n}\leqslant 2^{2r+1}(n+r-1)^{2r}$，(a) 当$1\leqslant n\leqslant 16$时，精确成立；(b)当$n\to\infty$时，渐近成立。

49. [HM25]请证明：当$n\to\infty$时，几乎所有的布尔函数 $f(x_1,\cdots,x_n)$都有最小式子长度$L(f)>2^n/\lg n-2^{n+2}/(\lg n)^2$。

50. [24] 什么是素数函数(35)的素蕴涵数和素数从句？请表示(a)DNF；(b)CNF中最小长度的函数。

51. [20]如果真值表的行是基于x_5x_6而不是x_1x_2，代替(37)的素数探测器的表示是怎样的？

52. [23]当$5\leqslant n\leqslant 16$，k和l为何数时，式(38)的上界可最小化？

53. [HM22]当 $k=\lfloor 2\lg n \rfloor$，$l=\lceil 2^k/(n-3\lg n) \rceil$，$n\to\infty$时，估计(38)的值。

54. [29]请找出一个短的布尔链来计算所有6个函数$f_j(x)=[x_1x_2x_3x_4\in A_j]$，其中$A_1=\{0010,0101,1011\}$，$A_2=\{0001,1111\}$，$A_3=\{0011,0111,1101\}$，$A_4=\{1001,1111\}$，$A_5=\{1101\}$，$A_6=\{0101,1011\}$。（这6个函数出现在素数探测器（37）中。）请比较你的链和Lupanov的一般方法中的最小项优先计算策略。

55. [34]请证明6位的素数检测函数的代价最多是14。

▶56. [16]请解释为什么表3中所有有14个或者更多的“无需关心”的函数的代价都是0。

57. [19]当(45)中$(x_1x_2x_3x_4)_2>9$时，7段“数字”显示的是什么？

▶58. [30] 一个4×4位的S-box是4位向量{0000,0001,…,1111}的一个排列；这样的排列被用作著名的密码系统（例如俄罗斯标准GOST 28147(1989)）的组成部分。每个4×4位的S-box与一个4函数的序列$f_1(x_1,x_2,x_3,x_4),\cdots f_4(x_1,x_2,x_3,x_4)$相关，它能做如下转换：$x_1x_2x_3x_4 \mapsto f_1f_2f_3f_4$。

请找出所有4×4位的S-box，其中$C(f_1)=C(f_2)=C(f_3)=C(f_4)=7$。

59. [29] 一个满足习题58的条件的S-box有(0,…,f)$\mapsto$(0,6,5,b,3,9,f,e,c,4,7,8,d,2,a,1)；换句话说，(f_1,f_2,f_3,f_4)的真值表分别是(179a,63e8,5b26,3e29)。请找出一个布尔链，使之能在少于20步内计算出这4个“最困难”函数。

60. [23] (Frank Ruskey)设$z=(x+y) \bmod 3$，其中$x=(x_1x_2)_2$，$y=(y_1y_2)_2$，$z=(z_1z_2)_2$，并且每个2位值只能是00，01或者10。请在6个布尔步骤内由x_1,x_2,y_1和y_2计算出z_1和z_2。

61. [34]接习题60，找到一个好的方式来计算$z=(x+y) \bmod 5$，通过使用这些3位值000,001,010,011,100。

62. [HM23] 考虑一个随机的有2^nc个“关心”和2^nd个“无需关心”的n变量布尔部分函数，其中$c+d=1$。请证明几乎所有这样的部分函数的代价都超过$2^nc/n$。

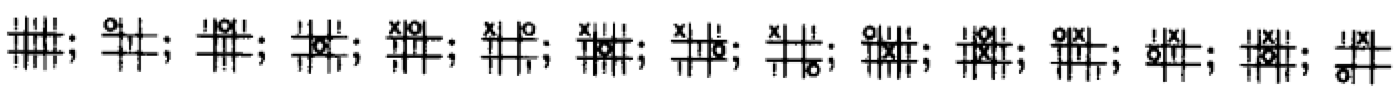
63. [HM35] (L. A. Sholomov, 1969) 接习题62，请证明所有这样的函数有$\leqslant 2^nc/n(1+O(n^{-1}\log n))$的代价。提示：存在一个$2^m(1+k)$个向量$x_1\cdots x_k$的集合，它与$k$立方体的每个$(k-m)$维子立方体相交。

64. [25] (魔力15) 两个游戏者轮流地选择从1到9的数字，每个数字只能选1次；如果存在胜利者的话，那么胜利者是第一个拿到三个加起来等于15的数字的人。有没有好的策略来赢得这个游戏？

▸65. [35]修改(47)～(56)中的tic-tac-toe的策略，使得它能够一直正确地工作。

66. [20]评价习题65中选择的移动。它们一直是最优的吗？

▸67. [40]与其为tic-tac-toe的每个位置简单地找到一个正确的移动，我们不如一下找到所有的。换句话说，给定了$x_1\cdots x_9o_1\cdots o_9$时，我们可以尝试计算出9个输出$g_1\cdots g_9$，其中$g_j=1$当且仅当移动到格子$j$的行动是属于×中最好的。例如，感叹号标识出对×在下列典型的位置上的所有正确的移动。



一个可以随机从这些可能的策略中选择的机器，相比只有一种固定策略的机器更为有趣。

一种能提供所有步骤的良好移动的有效策略是基于tic-tac-toe有8种对称的事实。想象一个芯片有18个输入$x_1\cdots x_9o_1\cdots o_9$和3个输出$(c,s,m)$，对于“角”、“边”和“中部”，芯片有性质：所需的函数g_j能够通过合适地组合芯片中的8个来实现。

$$\begin{aligned}
g_1&=c(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\quad\vee c(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9),\\
g_2&=s(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\quad\vee s(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7),\\
g_3&=c(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7)\\
&\quad\vee c(x_3x_6x_9x_2x_5x_8x_1x_4x_7o_3o_6o_9o_2o_5o_8o_1o_4o_7),\\
g_4&=s(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9)\\
&\quad\vee s(x_7x_4x_1x_8x_5x_2x_9x_6x_3o_7o_4o_1o_8o_5o_2o_9o_6o_3),\cdots\\
g_9&=c(x_9x_8x_7x_6x_5x_4x_3x_2x_1o_9o_8o_7o_6o_5o_4o_3o_2o_1)\\
&\quad\vee c(x_9x_6x_3x_8x_5x_2x_7x_4x_1o_9o_6o_3o_8o_5o_2o_7o_4o_1)
\end{aligned}$$

其中g_5是所有8个芯片的m个输出的或。

请使用少于2000个门来设计这样的一个芯片。

68. [M25]考虑n位π函数$\pi_n(x_1\cdots x_n)$，它的值是π的二进制表示中最重要位的右边第$(x_1\cdots x_n)_2$位。4.3.1节习题39中的方法描述了一种有效的方法来计算π的任意位，那么这种方法证明了对于足够大的n，都有$C(\pi_n)<2^n/n$吗？

69. [M24]设f的多线性表示是$\alpha_{000}\oplus\alpha_{001}x_m\oplus\alpha_{010}x_j\oplus\alpha_{011}x_jx_m\oplus\alpha_{100}x_i\oplus\alpha_{101}x_ix_m\oplus\alpha_{110}x_ix_j\oplus\alpha_{111}x_ix_jx_m$，其中每个系数$\alpha_l$是变量$\{x_1,\cdots,x_n\}\backslash\{x_i,x_j,x_m\}$的一个函数。

a) 请证明函数对(58)是"好"的，当且仅当系数满足$\alpha_{010}\alpha_{101}=\alpha_{011}\alpha_{100}$，$\alpha_{101}\alpha_{110}=\alpha_{100}\alpha_{111}$和$\alpha_{110}\alpha_{011}=\alpha_{111}\alpha_{010}$。

b) 当$f=(\det X)\bmod 2$时，(i,j,m)取何值会让函数对是"坏"的？（参见(60)。）

▸70. [M27]设X是3×3的布尔矩阵(60)。请为下列布尔函数找到有效的链：(a)$(\det X)\bmod 2$；(b) $[\mathrm{per}X>0]$；(c) $[\det X>0]$。

▸71. [M26]设$f(x)$在每个点$x=x_1\cdots x_n$等于0的可能性是p，独立于它在其他点的值。

a) 函数对(58)是好的的可能性是多少？

b) 坏的函数对(58)存在的可能性是多少？

c) 坏的函数对(58)在最多t次随机尝试中可以找到的可能性是多少？

d) 测试用例(i,j,m)对于p、t、n的函数的预期时间是多少？

72. [M24]将前一个习题扩展到部分函数的情况，其中$f(x)=0$的概率是p, $f(x)=1$的概率是q, $f(x)=*$的概率是r。

▸73. [20]如果对于所有满足$m\neq i\neq j\neq m$的(i,j,m)存在坏的函数对(58)，请证明在仅仅测试$\binom{n}{2}$个精心选择的三元组(i,j,m)后，f的不可分解性能够推导出来。

74. [25]扩展前一个习题的想法，当使用Shen、McKellar和Weiner的方法时，请推荐一种策略来选择连续的三元组(i,j,m)。

75. [20]当正文中的分解过程应用到"全-相等"函数$S_{0,n}(x_1,\cdots,x_n)$时会发生什么？

▸76. [M25](D. Uhlig, 1974) 这道习题的目的是证明一个有趣的事实：对于特定的函数f，用来计算布尔函数的最好的链$F(u_1,\cdots,u_n,\ v_1,\cdots,v_n)=f(u_1,\cdots,u_n)\vee f(v_1,\cdots,v_n)$的代价少于$2C(f)$；因此函数的分解并不是所有情况下都是一个好主意。

我们设$n=m+2^m$, $f(i_1\cdots,i_m,x_0,\cdots,x_{2^m-1})=f_i(x)$，其中$i$代表数$(i_1\cdots i_m)_2$。然后$(u_1,\cdots,u_n)=(i_1,\cdots,i_m,x_0,\cdots,x_{2^m-1})$, $(v_1,\cdots,v_n)=(j_1,\cdots j_m,y_0,\cdots,y_{2^m-1})$, $F(u,v)=f_i(x)\vee f_j(y)$。

a) 请证明代价为$O(n/\log n)^2$的链足够由给定的向量i、j、x和y来计算2^m+1个函数$z_l=x\oplus(([l\leqslant i]\oplus[i\leqslant j])\wedge(x\oplus y))$，其中$0\leqslant l\leqslant 2^m$；每个$z_l$是一个长度为$2^m$的向量。

b) 设$g_i(x)=f_i(x)\oplus f_{i-1}(x)$, 其中$0\leqslant i\leqslant 2^m$，$f_{-1}(x)=f_{2^m}(x)=0$。请估计计算$2^m+1$个值$c_l=g_l(z_l)$的代价，其中向量$z_l$对于$0\leqslant l\leqslant 2^m$的值是给定的。

c) 设$c'_l=c_l\wedge([i\leqslant j]\equiv[l\leqslant i])$，$c''_l=c_l\wedge([i\leqslant j]\equiv[j\leqslant l])$。请证明

$$f_i(x)=c'_0\oplus c'_1\oplus\cdots\oplus c'_{2^m},\ f_j(y)=c''_0\oplus c''_1\oplus\cdots\oplus c''_{2^m},$$

d) 请推导出$C(F)\leqslant 2^n/n+O(2^n(\log n)/n^2)$。（当$n$足够大的时候，代价肯定小于$2^{n+1}/n$，但是对于函数$f$有$C(f)>2^n/n$。）

为了清楚起见，为F写出当$m=1$和$f(i,x_0,x_1)=(i\wedge x_0)\vee x_1$时的链。

▸77. [35](N. P. Red'kin, 1970)假设一个布尔链只使用操作AND、OR或者NOT；因此，每一步或者是$x_i=x_{j(i)}\wedge x_{k(i)}$，或者是$x_i=x_{j(i)}\vee x_{k(i)}$，或者是$x_i=\bar{x}_{j(i)}$。请证明如果这样一个链要么计算"奇校验"函数$f_n(x_1,\cdots,x_n)=x_1\oplus\cdots\oplus x_n$，要么计算偶校验函数

$\overline{f}_n(x_1,\cdots,x_n)=1\oplus x_1\oplus\cdots\oplus x_n$， 其中$n\geqslant 2$，那么这个链的长度至少是$4(n-1)$。

78. [26](W. J. Paul, 1977) 设$f(x_1,\cdots,x_m,y_0,\cdots,y_{2^m-1})$是在$(x_1\cdots x_m)_2=k\in S$时任意一个等于$y_k$的布尔函数，其中一些给定的 $S\subseteq\{0,1,\cdots,2^m-1\}$；我们并不关心在其他点的$f$的值。请证明当$S$非空时$C(f)\geqslant 2\|S\|-2$。(特别地，当$S=\{0,1,\cdots,2^m-1\}$时，习题39的多路复用器链是渐进最优的。)

79. [32](C. P. Schnorr,1976) 我们称在布尔链中的变量u和v是“伙伴”，若在相关的二叉树图中只有一条简单路径在这两个变量之间。两个变量能够成为“伙伴”，仅当它们每个都在链中只使用一次；但是这个必要条件不是充分条件。例如，变量2和4是图9中$S_{1,2,3}$的链中的“伙伴”，但是它们不是S_2的链中的伙伴。

a) 请证明有n个变量而没有伙伴的布尔链的代价$\geqslant 2n-2$。

b) 请证明当f是全相等函数$S_{0,n}(x_1,\cdots,x_n)$时，有$C(f)=2n-3$。

▸ 80. [M27] (L. J. Stockmeyer, 1977) 对称函数的另一个表示法有时候显得很简便：如果$\alpha=a_0a_1\cdots a_n$是任意的二进制字符串，那么设$S_\alpha(x)=a_{\nu x}$。例如，在这个表示法里$\langle x_1x_2x_3\rangle=S_{0011}$，$x_1\oplus x_2\oplus x_3=S_{0101}$。注意$S_\alpha(0,x_2,\cdots,x_n)=S_{\alpha'}(x_2,\cdots,x_n)$和$S_\alpha(1,x_2,\cdots,x_n)=S_{'\alpha}(x_2,\cdots,x_n)$,其中$\alpha'$和$'\alpha$分别代表$\alpha$的最后和第一个元素被删除。而且，当$f$是任意$n-2$个变量的布尔函数时，有$S_\alpha(f(x_3,\cdots,x_n),\overline{f}(x_3,\cdots,x_n),x_3,\cdots,x_n)=S_{'\alpha'}(x_3,\cdots,x_n)$。

a) 一个奇偶性函数有$a_0\neq a_1\neq a_2\neq\cdots\neq a_n$。假定$n\geqslant 2$，请证明如果$S_\alpha$不是奇偶性函数，而$S_{'\alpha'}$非不变，那么$C(S_\alpha)\geqslant\max(C(S_{\alpha'})+2,\ C(S_{'\alpha})+2,\ \min(C(S_{'\alpha})+3,\ C(S_{'\alpha})+3,\ C(S_{'\alpha'})+5))$。

b) 当$0\leqslant k\leqslant n$时，从这个结果得出$C(S_k)$和$C(S_{>k})$的下界是多少？

81. [23] (M. Snir, 1986) 请证明习题36的前缀问题中的任意代价为c、深度为d的链都有$c+d\geqslant 2n-2$。

▸ 82. [M23] 请解释逻辑语句(62)～(70)。它们中的哪些是正确的？

83. [21] 如果对于$f(x_1,\cdots,x_n)$存在布尔链包含p个渠限化（canalizing）操作，那么请证明$C(f)<(p+1)(n+p/2)$。

84. [M20] 单调布尔链是每个操作符$\circ_i$都单调的布尔链。对于f的最短单调链的长度由$C^+(f)$来表示。如果对于$f(x_1,\cdots,x_n)$存在一个单调布尔链包含p个$\wedge$和q个$\vee$，请证明$C^+(f)<\min((p+1)(n+p/2),(q+1)(n+q/2))$。

▸ 85. [M28]设M_n是所有n变量单调函数的集合。如果L是包含在M_n中的一组函数，设

$$x\sqcup y=\bigwedge\{z\in L\mid z\supseteq x\vee y\}\ 和\ x\sqcap y=\bigvee\{z\in L\mid z\subseteq x\wedge y\}$$

如果L包含不变函数0和1以及投影函数x_j，其中$1\leqslant j\leqslant n$，并且如果当$x,y\in L$有$x\sqcup y\in L$，$x\sqcap y\in L$，那么我们称L“合法”。

a) 当$n=3$时我们能写$M_3=\{$`00,01,03,05,11,07,13,15,0f,33,55,17,1f,37,57,3f,5f,77,7f,ff`$\}$，用每个函数的16进制真值表来代替每个函数。共存在2^{15}组的L使得$\{$`00,0f,33,55,ff`$\}\subseteq L\subseteq M_3$；它们中的哪些是合法的呢？

b) 如果A是$\{1,\cdots,n\}$的一个子集，设$\lceil A\rceil=\bigvee_{a\in A}x_a$；同时设$\lceil\infty\rceil=1$。假设$A$是$\{1,\cdots,n\}$的子集的一个组，它包含所有$size\leqslant 1$的集合并在交集下是闭合的，换句话

说，当$A\in\mathcal{A}$且$B\in\mathcal{A}$时，$A\cap B\in\mathcal{A}$。证明组$L=\{\lceil A\rceil | A\in\mathcal{A}\cup\{\infty\}\}$是合法的。

c) 设$(x_{n+1},\cdots,x_{n+r})$是一个单调布尔链（1）。假设$(\hat{x}_{n+1},\cdots,\hat{x}_{n+r})$由同样的布尔链获得，只是每个操作符$\wedge$被改成了$\sqcap$，$\vee$改成了$\sqcup$，并与一些合法的组$L$相关。证明，对于$n+1\leqslant l\leqslant n+r$，我们肯定有

$$\hat{x}_l\subseteq x_l\vee\bigvee_{i=n+1}^{l}\{\hat{x}_i\oplus(\hat{x}_{j(i)}\vee\hat{x}_{k(i)})\mid\circ_i=\vee\}$$

$$\hat{x}_l\subseteq\hat{x}_l\vee\bigvee_{i=n+1}^{l}\{\hat{x}_i\oplus(\hat{x}_{j(i)}\wedge\hat{x}_{k(i)})\mid\circ_i=\wedge\}$$

86. [HM37]一个由点$\{1,\cdots,n\}$构成的图G能够由$N=\binom{n}{2}$个布尔变量x_{uv}定义，其中$1\leqslant u<v\leqslant n$，$x_{uv}=[G中u—v]$。设$f$是函数$f(x)=[G包含一个三角形]$；例如，当$n=4$时，$f(x_{12},x_{13},x_{14},x_{23},x_{24},x_{34})=(x_{12}\wedge x_{13}\wedge x_{23})\vee(x_{12}\wedge x_{14}\wedge x_{24})\vee(x_{13}\wedge x_{14}\wedge x_{34})\vee(x_{23}\wedge x_{24}\wedge x_{34})$。这道习题的目的就是证明单调复杂性$C^{+}(f)$是$\Omega(n/\log n)^3$。

a) 如果在图G中对$1\leqslant j\leqslant r$有$u_j—v_j$，那么将$S=\{\{u_1,v_1\},\cdots,\{u_r,v_r\}\}$称作一个$r$组，并设$\Delta(S)=\bigcup_{1\leqslant i<j\leqslant r}(\{u_i,v_i\}\cap\{u_j,v_j\})$是它的两两交集的元素。如果对于一些$r$组$S$，当$\Delta(s)\subseteq\{u,v\}$时有$u—v$，那么称$G$是$r$闭合的。除此之外，如果对于所有的$r$族$S$都有$|\Delta(S)|\geqslant 2$，那么它是强$r$闭合的。请证明一个强$r$闭合的图也是强$(r+1)$闭合的。

b) 请证明当$r>\max(m,n)$时，完全二分图$K_{m,n}$是强r闭合的。

c) 请证明一个强r闭合的图有至多$(r-1)^2$条边。

d) 设L是函数$\{1\}\cup\{\lceil G\rceil\mid G$是一个在$\{1,\cdots,n\}$上的强$r$闭合图$\}$构成的组。（参见习题85(b)；我们将$G$看作边的集合。例如，当这些边是1—3,1—4,2—3,2—4时，我们有$\lceil G\rceil=x_{13}\vee x_{14}\vee x_{23}\vee x_{24}$。）$L$是合法的吗？

e) 设$x_{N+1},\cdots,x_{N+p+q}=f$是一个$p\wedge$-步和$p\vee$-步的单调布尔链，并考虑(d)中基于组$L$的修改过的链$\hat{x}_{N+1},\cdots,\hat{x}_{N+p+q}=\hat{f}$。如果$\hat{f}\neq 1$，证明$2(r-1)^3p+(r-1)^2(n-2)\geqslant\binom{n}{3}$。提示：使用习题85(c)中的第二个公式。

f) 进一步，如果$\hat{f}=1$，那么我们肯定有$r^2q\geqslant 2^{r-1}$。

g) 因此$p=\Omega(n/\log n)^3$。提示：设$r\approx 6\lg n$并应用习题84。

87. [M20]请证明当非单调操作允许时，习题86的三角函数有代价$C(f)=O(n^{\lg 7}(\log n)^2)=O(n^{2.81})$。提示：一个图有一个三角形，当且仅当它的邻接矩阵的立方体有一条非零的对角线。

88. [40]一条中位链与一条布尔链相似，但是它采取三者取其中的步骤$x_i=\langle x_{j(i)}x_{k(i)}x_{l(i)}\rangle$，其中$n+1\leqslant i\leqslant n+r$，而不是(1)中的二元操作。请对于所有自对偶的7变量单调布尔函数，研究它们的最优长度、深度和中位链的代价。对于$\langle x_1x_2x_3x_4x_5x_6x_7\rangle$的最短链是什么？

习题答案

以愚人的蠢话回答愚人，否则你也和他一样愚蠢。

——《谚语》26:4

第7节

1. 根据提示，我们希望第二个‘$4m-4$’紧跟在第一个‘$2m-1$’之后。所需的排列可以从前4个示例推出，用十六进制表示即为：231213，46171435623725，86a31b1368597a425b4279，ca8e531f1358ac7db9e6427f2469bd。[R. O. Davies，《Math. Gazette》 43 (1959)，253—255。]

2. 当且仅当n mod 4等于0或1的时候这样的排列才存在。这个条件是必要的，因为这里必须有偶数个奇数。同时这个条件又是充分条件，因为我们可以在上一题得出的解前面添加‘00’来构造解。

注：这个问题由Marshall Hall在1951年首次提出，次年由F. T. Leahy，Jr.在未发表的著作[《Armed Forces Security Agency report》343(1953年1月28日)]中给予解答。它被T. Skolem和T. Bang独立提出并解决，《Math. Scandinavica》5 (1957)，57—58。对于其他数的间隔，参见J. E. Simpson给出的完整解，《Discrete Math》44 (1983)，97—104。

3. 是的。例如，环(0072362435714165)就不能被打开。

4. b的第k次出现是在从左往右的第$\lfloor k\phi\rfloor$位置，a的第k次出现是在$\lfloor k\phi^2\rfloor$位置。显然有$\lfloor k\phi^2\rfloor-\lfloor k\phi\rfloor=k$，因为$\phi^2=\phi+1$（$\lfloor k\phi\rfloor$形成了$\phi$的“频谱”（spectrum），参见《CMath》的习题3.13)。

5. $\binom{2n}{2}$个可能的位置中有$2n-k-1$个满足上述条件。如果这个概率是独立的（但它们不是），则$2L_n$的值将是

$$\binom{2n}{2,2,\cdots,2}\prod_{k=1}^{n}\left((2n-1-k)\Big/\binom{2n}{2}\right)=\frac{(2n)!^2\,n(n-1)}{n!(2n)^{n+1}(2n-1)^{n+1}}$$

$$=\exp\left(n\ln\frac{4n}{e^3}+\ln\sqrt{\frac{\pi en}{2}}+O(n^{-1})\right)$$

6. (a)当乘积被展开，我们将得到一个具有$(2n-2)!/(n-2)!$个项的多项式，且每项都是$4n$次。对每一个朗福德对存在一个项$x_1^2\cdots x_{2n}^2$；每一个其他项至少有一个1次的变量。沿着x_1，…，$x_{2n}\in\{-1，+1\}$加起来，取消所有坏的项，好的项有2^{2n}个。添加一个额外的因子2，因为有$2L_n$个朗福德对（包括左右对换）。

(b) 设 $f_k=\sum_{j=1}^{2n-k-1}x_jx_{j+k+1}$ 是第k个因子的主要部分。我们可以按格雷码顺序遍历所有4^n种情况x_1，…，$x_{2n}\in\{-1，+1\}$(算法7.2.1.1L)，每次仅对一个x_j求反。x_j的

每次改变最多引起每个f_k的两次调整，因此格雷码的每步代价为$O(n)$。

我们不需要计算和的精确值；当2^N远超过$2^{2n+1}L_n$时，该和足够用于计算模2^N。当n=24时，则更好一些，可以计算模$2^{60}-1$，或者同时计算模$2^{30}-1$和$2^{30}+1$。利用f_k=k+1(modulo 2)，也可以节省$\lceil n/2 \rceil$位精度。

(c) 第三个等号仅当n mod 4等于0或3时才成立；但是那些是有趣的n。和可以在n步内计算出来，第p步($p<n$)包含这些情况$x_{n-1}=x_{n+2}$，$x_{n-2}=x_{n+3}$，$x_{n-p+1}=x_{n+p}$，$x_{n-p}=x_n=x_{n+1}=+1$和$x_{n+p+1}=-1$；它有一个外部的循环以所有2^{p-1}种方式选择(x_{n-p+1}，…，x_{n-1})和一个内部循环以所有$2^{2n-2p-2}$种方式选择(x_1，…，x_{n-p-1}，x_{n+p+2}，…，x_{2n})。(内部循环使用格雷二进制码，推荐使用“风管琴顺序”（organ-pipe order）对下标进行优先级排序，从而使x_1和x_{2n}变化速度最快。外部循环则不必特别地高效。）第n步使用$x_j=x_{2n+1-j}$($1\leqslant j<n$且$x_n=x_{n+1}=+1$)覆盖2^{n-1}个回文的情况。如果S_p指代第p步的和，则$s_1+\cdots+s_{n-1}+s_n/2=2^{2n-2}L_n$。

这些项的大部分最终会变成0。例如，当n=16时，0的次数出现占了76%($2^{29}+2^{14}$个情况中的408 838 754个)。这个事实可以用来避免内循环中过多的乘法。(仅f_1，f_3，…可以是0。)

7. 设d_k是读取k个字符后不完整对的数目，因此$d_0=d_{2n}=0$，且对$1\leqslant k\leqslant 2n$有$d_k=d_{k-1}\pm 1$。d_k不超过6的最大的这种序列是(0, 1, 2, 3, 4, 5, 6, 5, 6…5, 6, 5, 4, 3, 2, 1, 0)。这个序列满足$\sum_{k=1}^{2n} d_k = 11n-30$。而任意朗福德对满足$\sum_{k=1}^{2n} d_k = \sum_{k=1}^{n}(k+1) = \binom{n+1}{2}+n$。因此$\binom{n+1}{2}+n\leqslant 11n-30$且$n\leqslant 15$。(事实上，当$n$=15时宽为6是不可能的。最大和最小的可能宽度一般情况下是未知的。)

8. 当n=4或n=7的时候无解。当n=8的时候有4个解：



(这个问题带来了一个有意思的机械类难题：使用宽k+1和高$\lceil k/2 \rceil$的部件构造第k件。在C. Dudley Langford最初的笔记[《Math. Gazette》42 (1958)，228]中阐述了类似的构造问题，且对n=12的情况提供了一个平面图的解。这个问题可以被转换成一个精确覆盖问题，用非初始列代表两个小配件不允许相交；参见7.2.2.1节习题00。Jean Brette基于Skolem的这个问题的变体设计了一个某种程度上类似的难题，使用宽度代替平面性；他在1992年抄送了一个备份给David Singmaster。)

9. 仅三种方式：181915267285296475384639743，191218246279458634753968357，191618257269258476354938743（以及它们的逆序）。[由G. Baron在1969年第一次发现；参见《Combinatorial Theory and Its Applications》(Budapest: 1970)，81—92。给定一个有132行的精确覆盖问题，7.2.2.1节的“跳舞链接”方法使用一棵仅含360个节点的搜索树解决了这个问题。]

10. 例如，设A=12，K=8，Q=4，J=0，♠=4，♡=3，♢=2，♣=1；加起来。

[这里，等价于图1的正交拉丁方阵在中世纪伊斯兰教的法器上已经隐式存在，并由Ibn al-Hajj在他的《Kitab Shumus al-Anwar》(*Cairo*: 1322)中给予了阐述；他还给出了5 × 5的示例。参见E. Doutté，《Magieet Religion dans l'Afrique du Nord 》(Algiers: 1909)，193—194，214，247；W. Ahrens，Der Islam 7 (1917)，228—238。也参见Lars D. Andersen的一篇讲述拉丁方阵历史的文章。]

11. $\begin{pmatrix} d\gamma\aleph & a\delta\beth & b\beta\gimel & c\alpha\daleth \\ c\beta\beth & b\alpha\aleph & a\gamma\daleth & d\delta\gimel \\ a\alpha\gimel & d\beta\daleth & c\delta\aleph & b\gamma\beth \\ b\delta\daleth & c\gamma\gimel & d\alpha\beth & a\beta\aleph \end{pmatrix}$。[Joseph Sauveur在《Mémoires de I'Académie Royale des Sciences》(Paris，1710)，92—138，§83中提出了这个已知最早的示例方阵。]

12. 如果n为奇数，我们可以设$M_{ij} = (i-j) \bmod n$。但是如果n是偶数，这里就没有横截：因为如果$\{(t_0+0) \bmod n, \cdots, (t_{n-1}+n-1) \bmod n\}$是一个横截，则我们有$\sum_{k=0}^{n-1} t_k \equiv \sum_{k=0}^{n-1}(t_k+k)(\text{modulo } n)$，因此$\sum_{k=0}^{n-1} k = \frac{1}{2}n(n-1)$是$n$的倍数。

13. 将每一个元素l替换为$\lfloor l/5 \rfloor$，可得一个由0和1组成的矩阵。记这4部分为$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$；则$A$和$D$均包含$k$个1，而$B$和$C$均包含$k$个0。假设原始方阵有10个分离的横截。如果$k \leqslant 2$，它们中最多有4个穿越$A$或$D$中的一个1，且最多有4个穿越$B$或$C$中的一个0。因此它们中至少有两个仅覆盖$A$和$D$中的0与$B$和$C$中的1。但是这样的一个横截上有偶数个0（不是5个），原因是它和A、D均等相交。

类似地，一个有正交阵的阶为4m+2的拉丁方阵，考虑了所有元素的换名之后，在它的每一个(2m+1) × (2m+1)子矩阵中必须有超过m个干扰者。[H. B. Mann，《Bull. Amer. Math. Soc》 (2) **50** (1944)，249—257。]

14. 问题(b)和(d)没有配偶。问题(a)、(c)和(e)分别有2、6和12265168(!)个解，依字典序，其第一个和最后一个解是

(a)	(a)	(c)	(c)	(e)	(e)
0456987213	0691534782	0362498571	0986271435	0214365897	0987645321
1305629847	1308257964	1408327695	1354068792	1025973468	1795402638
2043798165	2169340578	2673519408	2741853960	2690587143	2506913874
3289176504	3250879416	3521970846	3572690814	3857694201	3154067289
4518263790	4587902631	4890253167	4630789251	4168730925	4231850967
5167432089,	5412763890;	5736841920,	5218947306;	5473829016,	5348276190
6894015372	6945081327	6259784013	6095324178	6942158730	6820394715
7920341658	7836425109	7915602384	7869512043	7309216584	7069128543
8731504926	8723196045	8147036259	8407136529	8531402679	8412739056
9672850431	9074618253	9084165732	9123405687	9786041352	9673581402

注：方阵(a)、(b)、(c)和(d)都是从π、e、γ和φ的十进制数字舍弃与完整拉丁方阵不一致的每个数而得到的。尽管不是真正的随机，它们也可能是一般的典型拉丁方阵，大体上这些方阵中的一半都有正交阵。Parker为了获得一个不寻常的大量横截而构造了方阵(e)；它有5504个横截。(欧拉研究了一个类似的6阶方阵，因此

“错过”发现一个10×10对的机会）。

15. Parker失望地发现14(e)方阵的所有正交配偶中没有一个与其他正交。之后，他和J. W. Brown、A. S. Hedayat [《J. Combinatorics，Inf. and System Sci》18 (1993)，113—115]共同发现两个10×10方阵，它们有4个公共的横截。[参见B. Ganter、R. Mathon和A. Rosa，《Congressus Numerantium》20 (1978)，383—398；22 (1979)，181—204。] 在追踪L. Weisner的一个想法时[《Canadian Math. Bull》6 (1963)，61—63]，作者偶然注意到一些方阵很接近一个相互正交的三元组：下面这个方阵正交于它的转置；它有5个以对角线为轴对称的横截，在单元$(0, p_0)$，…，$(9, p_9)$中，$p_0 \cdots p_9$=0132674598，2301457689，3210896745，4897065312和6528410937。这些横截几乎是不相交的：它们覆盖了49个单元。

$$L=\begin{pmatrix}0&2&3&4&5&6&7&8&9&1\\3&1&9&2&7&0&8&5&4&6\\6&5&2&8&1&3&9&4&0&7\\8&7&5&3&2&4&1&9&6&0\\1&6&8&9&4&7&3&0&2&5\\4&9&7&0&8&5&2&6&1&3\\5&0&4&7&9&8&6&1&3&2\\9&4&1&6&3&2&0&7&5&8\\7&3&6&1&0&9&5&2&8&4\\2&8&0&5&6&1&4&3&7&9\end{pmatrix}\perp\begin{pmatrix}0&3&6&8&1&4&5&9&7&2\\2&1&5&7&6&9&0&4&3&8\\3&9&2&5&8&7&4&1&6&0\\4&2&8&3&9&0&7&6&1&5\\5&7&1&2&4&8&9&3&0&6\\6&0&3&4&7&5&8&2&9&1\\7&8&9&1&3&2&6&0&5&4\\8&5&4&9&0&6&1&7&2&3\\9&4&0&6&2&1&3&5&8&7\\1&6&7&0&5&3&2&8&4&9\end{pmatrix}=L^{T}$$

B. D. McKay、A. Meynert和W. Myrvold [《J. Comb. Designs》15 (2007)，98—119]证明了有非平凡对称的10×10的拉丁方阵不存在两个相互正交的配偶。两两正交的三个拉丁方阵，已知对于所有阶$n>10$都是存在的[参见 S. M. P. Wang和R. M. Wilson，《Congressus Numerantium》21 (1978)，688；D. T. Todorov，《Ars Combinatoria》20 (1985)，45—47]。

16. 参见R. A. Brualdi和H. J. Ryser，《Combinatorial Matrix Theory》(Cambridge University Press，1991)，§8.2。

17. (a)设存在$3n$列r_j、c_j、$v_j(0 \leqslant j<n)$和n^2行；行(i, j)中在列r_i、c_j和v_l的位置为1，其中$l=L_{ij}$ $(0 \leqslant i,\ j<n)$。

(b)设存在$4n^2$列r_{ij}、c_{ij}、x_{ij}、$y_{ij}(0 \leqslant i,\ j<n)$和$n^3-n^2+n$行；行$(i,\ j,\ k)$中在列$r_{ik}$、$c_{jk}$、$x_{ij}$和$y_{lk}$的位置为1，其中$l=L_{ij}(0 \leqslant i, j, k<n)$且$(i=k$或$j>0)$。

18. 给定一个正交阵列$\boldsymbol{A}$，有行A_i $(1 \leqslant i \leqslant m)$。定义拉丁方阵$\boldsymbol{L}_i=(L_{ijk})$ $(1 \leqslant i \leqslant m-2)$，当$A_{(m-1)q}=j$且$A_{mq}=k$ $(0 \leqslant j,\ k<n)$时令$L_{ijk}=A_{iq}$。(q的值由j和k的值唯一确定。)交换阵列中的列不改变相应的拉丁方阵。

构造过程也可以反过来，从n阶的相互正交的拉丁方阵构造n阶的正交阵列。例如，在习题11中，我们设$a=\alpha=$ ℵ $=0$，$b=\beta=$ ב $=1$，$c=\gamma=\lambda=12$和$d=\delta=$ ד $=3$，则可得到

$$A=\begin{pmatrix}3&0&1&2&2&1&0&3&0&3&2&1&1&2&3&0\\2&3&1&0&1&0&2&3&0&1&3&2&3&2&0&1\\0&1&2&3&1&0&3&2&2&3&0&1&3&2&1&0\\0&0&0&0&1&1&1&1&2&2&2&2&3&3&3&3\\0&1&2&3&0&1&2&3&0&1&2&3&0&1&2&3\end{pmatrix}$$

（从数学的角度来说，正交阵列的概念比正交的拉丁方阵的概念更清晰，因为它将内在的对称性表现得更清楚。注意，例如，一个$n\times n$的矩阵L是一个拉丁方阵当且仅当它正交于两个特定的非拉丁方阵，即

$$L\perp\begin{pmatrix}1&1&\cdots&1\\2&2&\cdots&2\\\vdots&\vdots&\ddots&\vdots\\n&n&\cdots&n\end{pmatrix}\text{和 }L\perp\begin{pmatrix}1&2&\cdots&n\\1&2&\cdots&n\\\vdots&\vdots&\ddots&\vdots\\1&2&\cdots&n\end{pmatrix}$$

因此拉丁方阵、希腊−拉丁方阵、希伯来−希腊−拉丁方阵等，分别等价于深度为3、4、5、…的正交阵列。此外，这里考虑的正交阵列仅仅是一个更一般的n元$m\times\lambda n^t$阵列，其中t表示强度（strength）、λ表示索引（index）的特例（t=2且λ=1）。这个概念由C. R. Rao引入并发表在《Proc. Edinburgh Math. Soc》8 (1949), 119—125；参见《Orthogonal Arrays》一书，由A. S. Hedayat、N. J. A. Sloane和J. Stufken著(Springer, 1999)。）

19. 可以将列重排从而使第一行为$0^n1^n\cdots(n-1)^n$。然后对其他行的元素重新编号使得均以$01\cdots(n-1)$开头。除第一行外的所有行，其余列的元素必定是不同的。

为达到当$n=p$时的上限，设每一列由两个数字x和y索引，此处$0\leqslant x,\ y<p$，将数字y，x，$(x+y)\bmod p$，$(x+2y)\bmod p$，…，$(x+(p-1)y)\bmod p$放入每一列。例如，当p=5时我们得到如下的正交阵列，其等价于4个相互正交的拉丁方阵。

$$\begin{pmatrix}0000011111222223333344444\\0123401234012340123401234\\0123412340234013401240123\\0123423401401231234034012\\0123434012123404012323401\\0123440123340122340112340\end{pmatrix}$$

[当n是一个素数的幂并基于有限域$GF(p^e)$时，可使用和上述方法本质上相同的另一个方法；参见E. H. Moore，《American Journal of Mathematics》18 (1896), 264—303，§15(l)。这些阵列等价于有限的射影平面（projective planes）；参见Marshall Hall，Jr.，《Combinatorial Theory》(Blaisdell，1967), 第12和13章。]

20. 设$\omega=e^{2\pi i/n}$，且设$a_1\cdots a_{n^2}$和$b_1\cdots b_{n^2}$是在不同行的向量。则$a_1b_1+\cdots+a_{n^2}b_{n^2}=\sum_{0\leqslant j,k<n}\omega^{j+k}=0$因为$\sum_{k=0}^{n-1}\omega^k=0$。

21. (a)为表明相等或平行是一个等价关系，需要验证传递律成立：如果$L\parallel M$且$M\parallel N$且$L\neq N$，则一定有$L\parallel N$。否则根据(ii)必定存在一个点p有$L\cap N=\{p\}$；而p位于平行于M的两条不同的线上，和(iii)矛盾。

(b)设$\{L_1,\ \cdots,\ L_n\}$是一个平行线类，假定M是另一个等价类中的一条线。则每

个L_j和M相交于唯一的点p_j；且M上的每一点都是这种情况，因为根据(iii)对于一个等价类，一个点仅位于其中一条线上。因此M恰包含n个点。

(c) 已经观察到当有m个等价类时，每个点属于m条线。如果线L、M和N属于三个不同的类，则M和N上的点数和L所在的等价类里线的数目相同。因此，有一条包含n个点的公共线，事实上点的总数是n^2（当然n可以是无限的）。

22. 给定一个阶为n、深度为m的正交阵列A，将A的列当作点，定义一个有n^2个点和m个平行线类的几何网；等价类k的线j是A的第k行中出现符号j的那些列的集合。

所有有$m\geqslant 3$个等价类的有限几何网都以这种形式出现。但是仅有一个类的几何网是从点到不相交子集的划分。有$m=2$个类的几何网有nn'个点(x, x')，此处有n条线‘x为常量’在一个等价类中和n'条线‘x'为常量’在另一个等价类中 [更多信息参见R. H. Bruck，《Canadian J. Math》3 (1951), 94—107；《Pacific J. Math》13 (1963), 421—457] 。

23. (a)如果$d(x, y)\leqslant t$ 且$d(x', y)\leqslant t$ 且$x \neq x'$，则$d(x, x')\leqslant 2t$。因此一个码字之间的距离大于$2t$的编码允许最多纠正t个错误，尽管计算也许比较复杂，至少在理论上是可以肯定的。相反地，如果$d(x, x')\leqslant 2t$且$x\neq x'$，存在一个元素y使$d(x, y)\leqslant t$和$d(x', y)\leqslant t$；因此我们不能从收到的y唯一地重新构造出x。

(b, c)设$m = r + 2$，观察到b^2个b元m元组构成的集合中所有元素对的海明距离都大于等于$m-1$，当且仅当它形成一个深度为m的b元正交阵列的列。[参见S. W. Golomb、E. C. Posner，《IEEE Trans》IT-10 (1964), 196—208。编码理论的著作常常用符号$(n+r, b^n, d)_b$来表示距离为d的一个编码$C(b, n, r)$。因此一个深度为m的b元正交阵列本质上是一个$(m, b^2, m-1)_b$编码。]

24. (a)假定$x_j\neq x'_j$ $(1\leqslant j\leqslant l)$和$x_j = x'_j(l < j\leqslant N)$。如果$l=0$我们约定$x=x'$。否则考虑对应$m$条线的奇偶校验位通过点1。这些位中最多$l-1$位对应那些通过了点$\{2, \cdots, l\}$的线。因此$x'$有至少$m-(l-1)$次奇偶性改变，而$d(x, x')\geqslant l + (m-(l-1))=m+1$。

(b)设$l_{p1}, \cdots, l_{pm}$是通过点p的线的索引数。接收到消息$y_1\cdots y_{N+R}$之后，取$m+1$个“证据位”（witness bit）$\{y_{p0}, \cdots, y_{pm}\}$的众数之值来计算$x_p(1\leqslant p\leqslant N)$。此处$y_{p0}=y_p$且$y_{pk}=(y_{N+l_{pk}}+\Sigma\{y_j|j\neq p$且点$j$在线$l_{pk}$上$\})\bmod 2$，$(1\leqslant k\leqslant m)$。因为每一个收到的位$y_j$最多影响证明位中的一位，所以这种方法是可行的。

例如，在习题19的25点几何网里，假定每个码字的奇偶校验位$x_{26+5i+j}$对应行i的线j $(0\leqslant i\leqslant 5$且$0\leqslant j < 5)$；因此$x_{26} = x_1\oplus x_2\oplus x_3\oplus x_4\oplus x_5$，$x_{27} = x_6\oplus x_7\oplus x_8\oplus x_9\oplus x_{10}$，$\cdots$，$x_{55}=x_5\oplus x_6\oplus x_{12}\oplus x_{18}\oplus x_{24}$。对于消息$y_1\cdots y_{55}$，译码位$x_1$需要计算这7位的众数：$y_1$，$y_{26}\oplus y_2\oplus y_3\oplus y_4\oplus y_5$，$y_{31}\oplus y_6\oplus y_{11}\oplus y_{16}\oplus y_{21}$，$y_{36}\oplus y_{10}\oplus y_{14}\oplus y_{18}\oplus y_{22}$，$y_{41}\oplus y_9\oplus y_{12}\oplus y_{20}\oplus y_{23}$，$y_{46}\oplus y_8\oplus y_{15}\oplus y_{17}\oplus y_{24}$，$y_{51}\oplus y_7\oplus y_{13}\oplus y_{19}\oplus y_{25}$。[7.1.2节解释了怎样高效地计算众数函数（majority function）。注意，如果只想最多纠正两个错误，可以除去最后10位，如果想纠正一个错误，除去最后20位，剩下的也足够了。参见M. Y. Hsiao、D. C. Bossen和R. T. Chien，《IBM J. Research and Development》14 (1970)，390—394。]

25. 考虑{`l`，`e`，`a`，`s`，`t`}的变位词（参见习题5-21），可得方阵

```
stela
telas
elast
laste
astel
```

和它的行的旋转。这里telas是西班牙语的“纤维织物”；elast是一个前缀，意思是灵活的；laste是一个命令性的乔叟式动词。(当然每一个可以读出来的5个字母的单词都已经用来拼写或错误拼写某个地方的某个东西，在曾经的某个时候。)

26. “every night, young video buffs catch rerun fever forty years after those great shows first aired.” [Robert Leighton，《GAMES》 16，6 (1992年12月)，34，47。]

27. 当k=(1, 2, 3, 4, 5)分别为(0, 4, 163, 1756, 3834)；mamma和esses给出一个“满堂红”(full house)。

28. 是的，共有38对。最常见的解是needs（秩为180）和offer（秩为384)。仅有三种始终相差+1的情况(adder beefs, sheer tiffs, sneer toffs)。其他印象较深的有ghost hints和strut rusts。每对词中都有一个词是以s结尾的，但有4个例外，如robed spade。[参见Leonard J. Gordon，《Word Ways》 23 (1990)，59—61。]

29. 有18个回文，从level（秩为184）到dewed（秩为5688)。有34个镜像对“devil lived”、“knits stink”、“smart trams”、“faced decaf”。

30. 在SGB中有105个这样的词，first，below，floor，begin，cells，empty和hills是最常见的；abbey和pssst是按字母升序的第一个和最后一个。(如果你不喜欢pssst，倒数第二个是mossy)。字母按降序（reverse）排列的仅有37个词，从mecca到zoned，当然，wrong也是答案之一。

31. 中间的词是其他两个词的平均值，所以两端的词应该是模2同余的；这个条件减少了在词典中查询的数目到大概1/32。在WORDS(5757)有大概119个这样的三元组，但是在WORDS(2000)中仅有两组：marry, photo, solve; risky, tempo, vague。[《Word Ways》 25 (1992)，13—15。]

32. 唯一一个合理的常见例子貌似是peopleless。

33. chief，fight, right, which, ouija, jokes, ankle, films, hymns, known, crops, pique, quart, first, first, study, mauve, vowel, waxes, proxy, crazy, pizza。(想法是找到最常见的词，满足x后面是$(x+1) \bmod 26$，x=a(0)，x=b(1)，…，x=z(25)。我们同时最小化了插入距离，因此选择了bacon而不是更常见的black。另外没有找到这样一个词的时候，crazy仿佛是最合理的。参见《OMNI》 16, 8 (1994年5月)，94。)

34. 在每一类中最上面两个（和总数）是：pssst和pffft(2)，schwa和schmo(2)，threw和throw(36)，three和spree(5)，which和think(709)，there和these(234)，their和great(291)，whooo和wheee(3)，words和first(628)，large和since(376)，water和never(1313)，value和

radio(84)，would和could(460)，house和voice(101)，quiet和queen(25)，queue(1)，ahhhh和ankhs(4)，angle和extra(20)，other和after(227)，agree和issue(20)，along和using(124)，above和alone(92)，about和again(58)，adieu和aquae(2)，earth和eight(16)，eagle和ounce(8)，outer和eaten(42)，eerie和audio(4)，(0)，ouija和aioli(2)，(0)，(0)；years和every是被忽略的868个词中最常见的。[为填充剩下的三个空位，Internet上建议用ooops，ooooh，和ooooo。参见P. M. Cohen，《Word Ways》10 (1977)，221—223。]

35. 对于n=1,2,…,5757，考虑WORDS(n)。图例中的二分查找树根为s，当n达到978（stalk的秩）时才可能。下一个支持这种二分查找树结构的根字母是c，当n=2503（craze的秩）时它的子节点才有足够的分枝。之后的突破分别发生在当n=2730(bulks)，3999(ducky)，4230(panty)，4459(minis)，4709(whooo)，4782(lardy)，4824(herem)，4840(firma)，4924(ridgy)，5343(taxol)。

（当顶层查找树获得Horton-Strahler数4时，发生了突破；参见7.2.1.6节习题124。一种类似新式诗歌的有趣的词的集合，出现在当分枝方式是从右到左而不是从左到右的时候：black, slack, crack, track, click, slick, brick, trick, blank, plank, crank, drank, blink, clink, brink, drink。事实上，从右到左的分支组成了一个完整的有81个叶子的三分查找树：males, sales, tales, files, miles, piles, holes, …, tests, costs, hosts, posts。）

36. 记立方体的元素为$a_{ijk}(1 \leqslant i, j, k \leqslant 5)$，则对称的条件可描述为$a_{ijk} = a_{ikj} = a_{jik} = a_{jki} = a_{kij} = a_{kji}$。通常一个$n \times n \times n$立方体有$3n^2$个单词，固定两个坐标，使第三个坐标的范围为1到$n$即可得到；而对称性意味着我们仅需要$\binom{n+1}{2}$个单词。因此当$n$=5时所需的单词个数从75减少到15。[Jeff Grant能在《Oxford English Dictionary》中发现75个合适的单词；参见《Word Ways》11 (1978)，156—157。]

将(stove,event)换作(store,erect)或者(stole,elect)可获得另外两个立方体。

37. 图中最密集的部分称为“裸核”（bare core），包含顶点bares和cores，它们的度都为25。

38. tears→raise→aisle→smile；第二个词也可以是reals。[(11)中从tears到smile是Lewis Carroll最初给出的5字词例子中的一个。若Carroll得知使用有向规则会使从smile到tears的转换变得更难应该会很高兴，因为按该方向需要4步。]

39. 一定是生成子图，绝不是导出子图。

40. (a) 2^e，(b) 2^n，对每一个E或V的子集有一个。

41. (a) n=1和n=2；P_0是未定义的。(b)n=0和n=3。

42. G有65/2条边（因此它是不存在的）。

43. 有：前面三个都是和图2(e)同构的。[事实上，左边的图示是和出版物中Petersen图已知的最早形状是相同的：参见A. B. Kempe，《Philosophical Transactions》177 (1886)，1—70，尤其是在§ 59的图13。但是右边的图是一个平面的哈密顿图，且周长为4。

44. 所有自同构必须将一个拐角点（corner point）映射为另一个拐角点，因为三条不同的长为2的路径仅能在特定的非拐角点之间找到。因此该图仅有和C_4相同的8个对称关系。

45. 这个图的所有边仅连接来自同一行或相邻行的点。因此可以用颜色0和2在偶数行交替地着色，颜色1和3在奇数行交替地着色。NV的相邻点形成了一个5回路，因此4着色是必要的。

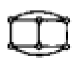
46. (a)每个顶点的度数都≥2，且由于入度的线使得它的邻居可以有定义良好的循环序。如果u—v且u—w，此处v、w是u相邻点循环序列中紧挨着的两个顶点，则一定有v—w。因此任何顶点u附近的所有点都属于一个唯一的三角形区域。

(b)这个公式当n=3时成立。如果n>3，收缩任意一条边为一个点；这个变换移除了一个顶点和三条边。（如果u—v被收缩，假定它是三角形x—u—v—x和y—u—v—y的一部分。我们失去了顶点v和边{x—v，u—v，y—u}；所有其他的边如w—v变成w—u。）

47. 它的一个平面上的图示将平面分成若干个区域，每个区域有4个或6个顶点在其边界上（因为$K_{3,3}$没有奇回路）。如果每一类的数量分别是f_4和f_6，则我们一定有$4f_4 + 6f_6 = 18$，因为该图有9条边；因此$(f_4, f_6) = (3, 1)$或者$(0, 3)$。我们也可以添加$f_4 + 3f_6$条边使其三角形化；但是那时图上至少有15条边，和习题46矛盾。

[$K_{3,3}$是非平面图这个事实可以追溯到一个谜题，要求连接三个房子到三个公共设施（水、煤气和电）而不能有交叉的管道。该谜题来源已不可考究；H. E. Dudeney 在《Strand》46 (1913), 110中称它为“前人的谜题”。]

48. 如果u、v、w是顶点且u—v，则有$d(w, u) \neq d(w, v)$ (modulo 2)；否则从w到u的最短路径和从w到v的最短路径将产生一个奇回路。在w着色为0之后，程序将所有和一个已着色的顶点u相邻的未着色的顶点v的颜色确定为$d(w, v)$ mod 2；在重新选择一个w之前的所有满足$d(w, v) < \infty$的顶点v都已被着色。

49. 仅有三个：K_4、$K_{3,3}$和

（即$\overline{C}_6$）。

50. 这个图一定得是连通的，因为3着色的数量可以被3^r整除，此处r为连通分量的数目。它还一定得是完全二分图$K_{m,n}$，可以用$3(2^m+2^n-2)$种方式三着色。从$K_{m,n}$中删除边并不减少着色的数量；因此$(2^m+2^n-2) \leq 8$，而我们有$\{m, n\}=\{1, 1\}$，$\{1, 2\}$，$\{1, 3\}$或$\{2, 2\}$。所以仅有的可能是爪（claw）$K_{1,3}$和路径P_4。

51. 一个4回路p_1—L_1—p_2—L_2—p_1对应有两个公共点$\{p_1, p_2\}$的两条不同的线$\{L_1, L_2\}$，这和(ii)矛盾。所以周长至少是6。

如果仅有一个平行线类，周长是∞；如果有两个类，对于$n \leq n'$个成员，周长为8，或者如果n=1，周长为∞。（参见习题22的答案。）否则我们可以从三个不同的类取三条线构成三角形，从而可找到一个长为6的回路。

52. 如果直径是d且周长是g，则$d \geqslant \lfloor g/2 \rfloor$，除非$g=\infty$。

53. happy（它连接着tears和sweat，但不连接world）。

54. (a) 它是单个、高度连通的分量。（顺便提一句，这个图是一个二分图的线图。这个二分图的一部分对应首字母{A，C，D，F，G，…，W}，另一部分对应末字母{A，C，D，E，H，…，Z}。）

(b) 顶点WY是孤立的。其他顶点，有些入度为0，即FL，GA，PA，UT，WA，WI和WV，它们自己构成强连通分量；它们都在一个庞大的强连通分量之前，其后是剩下的出度为0的单点连通分量：AZ，DE，KY，ME，NE，NH，NJ，NY，OH，TX。

(c) 现在强连通分量{GU}在{UT}之前；NH，OH，PA，WA，WI和WV加入到那个庞大的连通分量中；{FM}在它之前，{AE}和{WY}在它之后。

55. $\binom{N}{2}-\binom{n_1}{2}-\cdots-\binom{n_k}{2}$，此处$N=n_1+\cdots+n_k$。

56. 正确。注意J_n是简单的，但它不对应到任何多重图。

57. 错误，譬如有向连通图$u \to w \leftarrow v$。（但是u、v在同一个强连通分量中，当且仅当$d(u,v)<\infty$且$d(v,u)<\infty$；参见2.3.4.2节。）

58. 每个连通分量都是一个回路，它的阶至少是(a) 3 (b) 1。

59. (a) 在n上归纳，我们可以使用直接的插入排序：假设$v_1 \to \cdots \to v_{n-1}$。则有$v_n \to v_1$或者$v_{n-1} \to v_n$或者$v_{k-1} \to v_n \to v_k$，此处$k$是满足$v_n \to v_k$的最小的数。[L. Rédei，《Acta litterarum ac scientiarum》7 (Szeged, 1934)，39—43。]

(b) 15：01234，02341，02413和它们的循环移位。[这种有向路径的数目始终是奇数；参见T. Szele，《Matematikaiés Fizikai Lapok》50 (1943), 223—256。]

(c) 是的。（归纳法：如果这里仅有一个位置插入v_n，如(a)部分，这个竞赛图是传递的。）

60. 设$A=\{x|u \to x\}$，$B=\{x|x \to v\}$，$C=\{x|v \to x\}$。如果$v \notin A$且$A \cap B=\varnothing$，我们有$|A|+|B| = |A \cup B| \leqslant n-2$，因为$u \notin A \cup B$且$v \notin A \cup B$。但是$|B|+|C| = n-1$；因此$|A| < |C|$。[H. G. Landau,《Bull. Math. Biophysics》15 (1953)，148。]

61. $1 \to 1$，$1 \to 2$，$2 \to 2$；则$A=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$且对所有整数$k$有$A^k=\begin{pmatrix}1 & k\\ 0 & 1\end{pmatrix}$。

62. (a) 假设顶点是{1，…，n}，积和式展开有$n!$项，项$a_{1p_1}\cdots a_{np_n}$恰对应了由弧$\{1 \to p_1, \cdots, n \to p_n\}$组成的有向生成置换图。

(b) 使用类似的论据可以说明det A即是偶生成置换图的数目减去奇生成置换图的数目。[参见F. Harary，《SIAM Review》4 (1962)，202—210，其中有向置换图被称作“线性子图”。]

63. 设v是任一顶点。如果$g = 2t + 1$，至少$d(d-1)^{k-1}$个顶点x满足$d(v, x) = k$ $(1 \leqslant k < t)$。如果$g = 2t + 2$，v'是v的任一邻居，也有至少$(d-1)^t$个顶点x满足$d(v, x) = t+1$ 且$d(v', x) = t$。

64. 为达到习题63的下界，每一个顶点v的度数都必须为d，且v的d个邻居都必

须和剩下的$d-1$个顶点相邻。事实上，这个图即是$K_{d,d}$。

65. (a) 根据习题63的结论，G必须是度为d的正规图，且在两个不同的顶点之间一定恰有一条长度不超过2的路径。

(b) 不妨取$\lambda_1=d$，$x_1=(1\cdots1)^T$。所有其他的特征向量满足$Jx_j=(0\cdots0)^T$；因此，$\lambda_j^2+\lambda_j=d-1(1<j\leqslant N)$。

(c) 如果 $\lambda_2=\cdots=\lambda_m=(-1+\sqrt{4d-3})/2$ 和 $\lambda_{m+1}=\cdots=\lambda_N=\left(-1-\sqrt{4d-3}\right)/2$，则我们一定有$m-1=N-m$。由这个值我们发现$\lambda_1+\cdots+\lambda_N=d-d^2/2$

(d) 如果$4d-3=s^2$且m如(c)中描述，则特征值的和为

$$\frac{s^2+3}{4}+(m-1)\frac{s-1}{2}-\left(\frac{(s^2+3)^2}{16}+1-m\right)\frac{s+1}{2}$$

结果是15/32加上一个s的倍数。因此s必须是15的约数。

[这些结果来自A. J. Hoffman和R. R. Singleton，《IBM J. Research and Development》4 (1960)，497—504，文中也证明了$d=7$的图是唯一的。]

66. 用$[a, b]$和$(a, b)(0\leqslant a,b<5)$来指代50个顶点，用对5取模的算术运算定义三种边：

$[a,b]$—$[a+1,b]$；(a,b)—$(a+2,b)$；(a,b)—$[a+bc, c]$ 其中$0\leqslant a, b, c<5$

[参见W. G. Brown，《Canadian J. Math》19 (1967), 644—648；《J. London Math. Soc》42 (1967), 514—520。如果没有前面两种边，图的周长为6，对应习题51中的几何网，使用习题19的解答中的正交阵列。]

67. 所有确定的可能都已被Michael Aschbacher排除，参见《Journal of Algebra》19 (1971)，538—540。

68. 如果G有s个自同构，则它有$n!/s$个邻接矩阵，因为存在s个置换矩阵$\boldsymbol{P}$使$\boldsymbol{P}^{-}\boldsymbol{A}\boldsymbol{P}=\boldsymbol{A}$。

69. 首先对所有顶点v设置IDEG(v)←0。然后对所有v执行(31)，在该微算法的第二行设置u←TIP(a)和IDEG(u)←IDEG(u)+1。

如果要“对所有v”做某种操作，使用SGB格式，先设置v←VERTICES(g)；然后当v < VERTICES(g)+N(g)时，执行操作并设置v←v+1。

70. 步骤B1被执行一次（但是它使用$O(n)$单位时间）。步骤(B2, B3, …, B8)步分别被执行$(n+1, n, n, m+n, m, m, n)$次，每次代价为$O(1)$。

71. 可能有多种选择。这里我们使用32位指针，以相对位于Data_Segment的符号地址Pool。下面的描述提供了一种用于处理基本SGB数据结构的使用示例。

```
VSIZE IS 32; ASIZE IS 24                   节点规模
ARCS IS 0; COLOR IS 8; LINK IS 12          顶点域的偏移量
TIP IS 0; NEXT IS 4                        弧域的偏移量
arcs GREG Pool+ARCS; color GREG Pool+COLOR; link GREG Pool+LINK
tip GREG Pool+TIP; next GREG Pool+NEXT
u GREG; v GREG; w GREG; s GREG; a GREG; mone GREG -1
AlgB    BZ     n, Success                  如果图为空则退出
        MUL    $0, n, VSIZE                B1. 初始化
```

```
         ADDU   v, v0, $0          v←v0+n
         SET    w, v0              w←v0
1H       STT    mone, color, w     COLOR(w)←-1
         ADDU   w, w, VSIZE        w←w+1
         CMP    $0, w, v
         PBNZ   $0, 1B             重复直到w=v
0H       SUBU   w, w, VSIZE        w←w-1
3H       LDT    $0, color, w       B3.如果必要则对w着色
         PBNN   $0, 2F             如果COLOR(w)≥0则跳至B2
         STCO   0, link, w         COLOR(w)←0, LINK(w)←Λ
         SET    s, w               s←w
4H       SET    u, s               B4.Stack⇒u,设置u←s
         LDTU   s, link, s         s←LINK(s)
         LDT    $1, color, u
         NEG    $1, 1, $1          $1←1-COLOR(u)
         LDTU   a, arcs, u         a←ARCS(u)
5H       BZ     a, 8F              B5.是否完成u? 如果a=Λ跳至B8
5H       LDTU   v, tip, a          v←TIP(a)
6H       LDT    $0, color, v       B6.处理v
         CMP    $2, $0, $1         (此处程序使用了一点技巧)
         PBZ    $2, 7F             如果COLOR(v)=1-COLOR(u)则跳至B7
         BNN    $0, Failure        如果COLOR(v)=COLOR(u)则失败
         STT    $1, color, v       COLOR(v)←1-COLOR(u)
         STTU   s, link, v         LINK(v)←s
         SET    s, v               s←v
7H       LDTU   a, next, a         B7.在a上循环。设置a←NEXT(a)
         PBNZ   a, 5B              如果a≠Λ则跳至B5
8H       PBNZ   s, 4B              B8.栈非空? 如果s≠Λ则跳至B4
2H       CMP    $0, w, v0          B2.完成?
         PBNZ   $0, 0B             如果w≠v0, 减少w, 跳至B3
Success  LOC    @                  (成功结束)▮
```

72. (a) 当顶点进入或离开栈时，这个条件会保持不变。

(b) 顶点v已经被着色但还没有被浏览，因为每个已被浏览顶点的邻居都被着了合适的颜色。

(c) 在步骤B6设置$s \leftarrow v$之前，设置PARENT(v)←u，此处PARENT是一个新的效用域。在该步非成功结束之前，做了如下事情："重复输出NAME(u)和设置$u \leftarrow$ PARENT(u)，直到u=PARENT(v)；然后输出NAME(u)和NAME(v)。"

73. K_{10}.(且*random_graph*(10, 100, 0, 1,1,0, 0, 0, 0, 0)是J_{10}。)

74. badness有出度22；其他顶点的出度都不超过20。

75. 设参数(n_1, n_2, n_3, n_4, p, w, o)分别是(a) (n,0,0,0,−1,0,0); (b) (n,0,0,0,1,0,0); (c) (n,0,0,0,1,1,0); (d) (n,0,0,0,−1,0,1); (e) (n,0,0,0,1,0,1); (f) (n,0,0,0,1,1,1); (g) (m,n,0,0,1,0,0); (h) (m,n,0,0,1,2,0); (i) (m,n,0,0,1,3,0); (j) (m,n,0,0,−1,0,0); (k) (m,n,0,0,1,3,1); (l) (n,0,0,0,2,0,0); (m) (2,−n,0,0,1,0,0)。

76. 是的。例如习题75的解答(c)中从C_1到C_2。(但是p<0时不会出现自循环

(self-loop)，因为对$k=1,2,\cdots$会生成弧$x\to y=x+k\delta$，直到y超出范围或$y=x$。)

77. 假定x和y是顶点且$d(x,y)>2$。因此$x \not\!\!- y$；如果v是任何其他顶点，我们必定有$v \not\!\!- x$或$v \not\!\!- y$。由此可知在$\overline{G}$中任意两个顶点u和v之间的一条路径长度最多为3。

78. (a) 边的数目，$\binom{n}{2}\Big/2$一定是一个整数。最小的例子是K_0、K_1、P_4、C_5和。

(b) 如果q是任何奇数，我们有$u - v$当且仅当$\varphi^q(u) \not\!\!- \varphi^q(v)$。因此$\varphi^q$不能有两个固定点，也不能包含长度为2的回路。

(c) V的这样一个置换也定义了对K_n的边的一个置换$\hat{\varphi}$，即取$\{u,v\}\mapsto\hat{\varphi}$ ($\{u,v\}$)=$\{\varphi(u),\varphi(v)\}$，然后容易发现$\hat{\varphi}$的回路长度都是4的倍数。如果$\hat{\varphi}$有个$t$条回路，我们可以通过对回路中的边交替着色，从而获得2^t个自补图。

(d) 这种情况下φ有一个唯一的固定点v，且$G'=G\backslash v$是一个自补图。假定φ除(v)有r条回路，则$\hat{\varphi}$有r条回路包含连接顶点v的边，存在2^r种方式扩展G'为图G。

[参考：H. Sachs，《Publicationes Mathematicæ》9 (Debrecen, 1962), 270—288; G. Ringel，《Archiv der Mathematik》14 (1963), 354—358。]

79. 解法1：来自H. Sachs，设$\varphi=(1\ 2\ \cdots 4k)$，当$u>v>0$且$u+v \bmod 4\leqslant 1$时设置$u - v$；当$v \bmod 2=0$ 时设置$0 - v$。

解法2：设$\varphi=(a_1\ b_1\ c_1\ d_1)\cdots(a_k\ b_k\ c_k\ d_k)$，此处$a_j=4j-3$, $b_j=4j-2$, $c_j=4j-1$和$d_j=4j$。让$0 - b_j - a_j - c_j - d_j - 0 (1\leqslant j\leqslant k)$，且$a_i - a_j - b_i - d_j - c_i - c_j - d_i - b_j - a_i (1\leqslant i<j\leqslant k)$。

80. （解法来自G. Ringel）设φ为习题79的解法2中所定义的。E_0为$3k$条边$b_j - a_j - c_j - b_j$ $(1\leqslant j\leqslant k)$；$E_1$为$\{a_i,b_i,c_i,d_i\}$和$\{b_j,d_j\}$之间的$8\binom{k}{2}$条边$(1\leqslant i<j\leqslant k)$；$E_2$为$\{a_i,b_i,c_i,d_i\}$和$\{a_j,c_j\}$之间的$8\binom{k}{2}$条边$(1\leqslant i<j\leqslant k)$。在情况(a)中，$E_0\cup E_1$的直径为2；$E_0\cup E_2$的直径为3。(b)的情况类似，但是我们添加$2k$条边$b_j - 0 - d_j$到$E_1$中，添加$2k$条边$a_j - 0 - c_j$到$E_2$中。

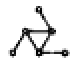
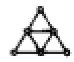
81. C_3^-，K_3^-，$D=$和$D^T=$。（有向图D的反转D^T表示反转它所有弧的方向。存在16个非自同构的3阶无环的简单有向图，其中10个是自反转的，包括C_3^-和K_3^-。）

82. (a)正确，由定义可知。(b)正确：如果每个顶点有d个邻居，每条边$u - v$有$d-1$个邻居$u - w$和$d-1$个邻居$w - v$。(c)正确：$\{a_i,b_j\}$有$m+n-2$个邻居，$0\leqslant i<m$且$0\leqslant j<n$。(d)错误：$L(K_{1,1,2})$有5个顶点和8条边。(e)正确。(f)正确：仅有的不相邻的边是$\{0,1\}\not\!\!-\{2,3\}$，$\{0,2\}\not\!\!-\{1,3\}$，$\{0,3\}\not\!\!-\{1,2\}$。(g)正确，对所有$n>0$。(h)错误，除非G没有孤立的顶点。

83. 它是Petersen图。[A. Kowalewski，《Sitzungsberichte der Akademie der Wissenschaften in Wien, Mathematisch-Nat. Klasse, Abteilung IIa》126 (1917), 67—90。]

84. 是的。设$\varphi(\{a_u,b_v\})=\{a_{(u+v)\bmod 3},b_{(u-v)\bmod 3}\}$，$0\leqslant u,v<3$。

85. 设顶点的度数为$\{d_1,\cdots,d_n\}$。则G有$\frac{1}{2}(d_1+\cdots+d_n)$条边，且$L(G)$有$\frac{1}{2}(d_1(d_1-1)+\cdots+d_n(d_n-1))$。因此$G$和$L(G)$都恰有$n$条边，当且仅当$(d_1-2)^2+\cdots+(d_n-2)^2=0$。最终习题58给出了答案。[参见V. V. Menon，《Canadian Math. Bull》8 (1965), 7—15。]

86. 如果$G=$ ，则$\overline{G}=$  $=L(G)$。

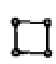
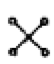
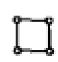
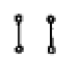
87. (a)是，非常容易。[事实上，R. L. Brooks已经证明了每个最大度数$d>2$的连接图，除了完全图K_{d+1}，都是d可着色的；参见《Proc. Cambridge Phil. Soc》37 (1941), 194—197。]

(b)否。实际只有一种方式对图2(e)的外层的5回路进行3着色；这引起在内层5回路上的冲突。[Petersen在1898年证明了这一点。]

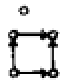
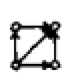
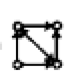
88. 一条回路不使用中心的顶点，有$(n-1)(n-2)$条回路要使用（即，一条回路恰对应边框上不同顶点的一个有序对）。我们不对C_0计算。

89. 分别地，等式两边都等于$\begin{pmatrix} A & O & O \\ O & B & O \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ J & B & J \\ J & J & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ O & B & J \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & O & O \\ J & B & O \\ J & J & C \end{pmatrix}$。

90. K_4和$\overline{K_4}$；$K_{1,1,2}$和$\overline{K_{1,1,2}}$；$K_{2,2}=C_4$和$\overline{K_{2,2}}$；$K_{1,3}$和$\overline{K_{1,3}}$；$K_1\oplus K_{1,2}$和它的补图；根据(47)所有图K_α是补图。$P_4=\overline{P_4}$不是补图。（一个补图的所有连通子图都有直径$\leqslant 2$；W_5是一个补图，但W_6不是。）

91. (a) ；(b) ；(c) ；(d) ；(e) ；(f) ；(g) 。（一般情况下，我们有$K_2\triangle H=(K_2\square H)\cup(K_2\oplus\overline{H})$和$K_2\circ H=H-H$。因此恰巧$K_2\triangle H=K_2\square H$和$=K_2\circ H=K_2\boxtimes H$会发生当且仅当$H$是一个完全图。）

记法说明：由J. Nešetřil在《Lecture Notes in Comp. Sci》118 (1981), 94—102中建议的记法$G\square H$和$G\boxtimes H$可很好地与图示(a)和(c)相符合。他的类似建议，即以$G\times H$表示(b)也很有吸引力；但这里没有采纳，因为已有上百个作者使用$G\times H$指代$G\square H$了。

92. (a) ；(b) ；(c) ；(d) ；(e) 。

93. $K_m\boxtimes K_n=K_m\circ K_n\cong K_{mn}$。

94. 否。它们是$K_{26}\square K_{26}\square K_{26}\square K_{26}\square K_{26}$的导出子图。

95. (a)d_u+d_v; (b) d_ud_v; (c) $d_ud_v+d_u+d_v$; (d) $d_u(n-d_v)+(m-d_u)d_v$; (e) d_un+d_v。

96. (a)$A\square B=A\oplus I+I\oplus B$。(b)$A\boxtimes B=A\square B+A\oplus B$。(c)$A\triangle B=A\oplus J+J\oplus B-2A\oplus B$。(d)$A\circ B=A\oplus J+I\oplus B$。（公式(a)、(b)和(d)定义了任意有向图和多重图的积。公式(c)对简单有向图是成立的；但是当A和B包含的值大于1时负的项可能会出现。）

历史注释：矩阵直积通常被称为Kronecker积，因为K. Hensel [《Crelle》105 (1889), 329—344]称这是他在Kronecker的讲座中听到的，然而事实上Kronecker从没出版过任何关于矩阵直积的著作。已知矩阵直积的第一次出现是在J. G. Zehfuss的论文中[《Zeitschrift für Math. und Physik》3 (1858), 298—301]，他证明

了当$m=m'$且$n=n'$时$\det(A\otimes B)=(\det A)^{n}(\det B)^{m}$。基本公式$(A\otimes B)^{T}=A^{T}\otimes B^{T}$，$(A\otimes B)(A'\otimes B')=AA'\otimes BB'$和$(A\otimes B)^{-1}=A^{-1}\otimes B^{-1}$是由于A. Hurwitz [《Math. Annalen》45 (1894), 381—404]而得来的。

97. 邻接矩阵上的操作证明了$(G\oplus G')\square H=(G\square H)\oplus(G'\square H)$；$(G\oplus G')\boxtimes H=(G\boxtimes H)\oplus(G'\boxtimes H)$；$(G\oplus G')\circ H=(G\circ H)\oplus(G'\circ H)$。因为$G\square H\cong H\square G$，$G\otimes H\cong H\otimes G$和$G\boxtimes H\cong H\boxtimes G$，也有右分配律$G\square(H\oplus H')\cong(G\square H)\oplus(G\square H')$；$G\otimes(H\oplus H')\cong(G\otimes H)\oplus(G\otimes H')$；$G\boxtimes(H\oplus H')\cong(G\boxtimes H)\oplus(G\boxtimes H')$。字典积（lexicographic product）满足$\overline{G\circ H}=\overline{G}\circ\overline{H}$；也有$K_m\circ H=H—\cdots—H$，因此$K_m\circ\overline{K_n}=K_{n,\cdots,n}$。更进一步有$G\circ K_n=G\boxtimes K_n$；$K_m\otimes K_n=\overline{K_m\square K_n}=L(K_{m,n})$。

98. 有kl个连通分量。（基于之前习题所证的分配律，以及当G和H是连通的，则$G\square H$和$G\boxtimes H$也是连通的这一事实。）

99. 在$G\square H$中从(u,v)到(u',v')的每条路径都必须使用至少$d_G(u,u')$“G步”和至少$d_H(v,v')$“H步”，且最小值是可达到的。类似的道理可以说明$d_{G\boxtimes H}((u,v),(u',v'))=\max(d_G(u,u'),d_H(v,v'))$。

100. 如果G和H是连通的，且如果它们至少分别有两个顶点，$G\otimes H$不连通当且仅当G和H是二分图。“当”的部分容易理解；反之，如果G中存在奇数长度的回路，我们可以用如下方法从(u,v)到(u',v')：首先到(u'',v')，这里u''是G任一可用的顶点。然后在G中走偶数步从u''到u'，同时在H中于v'和它的邻居之间来回移动。[P. M. Weichsel，《Proc Amer. Math. Soc》13 (1962), 47—52。]

101. 选择度数最大的顶点u和v。然后根据习题95有$d_u+d_v=d_ud_v$；因此$G=H=K_1$或$d_u=d_v=2$。在后一种情况，$G=P_m$或C_m，且$H=P_n$或C_n。但是$G\square H$是连通的，所以G或H必定不是二分图，比如说G。则$G\square H$不是二分的，所以H必定也不是二分的；因此$G=C_m$且$H=C_n$，其中m和n都是奇数。在$C_m\square C_n$中最短的回路长度为$\min(m,n)$；在$C_m\otimes C_n$中则长度为$\max(m,n)$；因此$m=n$。反之，如果$n\geqslant 3$且是奇数，我们有$C_n\square C_n\cong C_n\otimes C_n$，其同构为映射$(u,v)\mapsto((u+v)\bmod n,(u-v)\bmod n)$，对于$0\leqslant u,v<n$。[D. J. Miller，《Canadian J. Math》20 (1968), 1511—1521。]

102. $P_m\boxtimes P_n$。（只有$\min(m,n)\leqslant 2$或$m=n=3$时它才是平面的。）

103.

1	2	3	4	5	7
2	1	3	4	6	8
3	1	2	5	6	8
4	1	2	5	6	
5	3	4	1	7	
6	2	3	4		
7	5	1			
8	2	3			

1	2	3	4	5	6	7	8	9
2	1	3	4	6	8	9		
3	1	2	5	6	8	9		
4	1	2	5	7				
5	3	4	1	7				
6	2	3	1	7				
7	4	5	6	1				
8	2	3	1	9				
9	8	2	3	1				

104. 为保持表的形状，边必须按照迂回的顺序生成。变量r和i分别记为第t列的起始行和结束行。例如，习题103的第二部分从$i\leftarrow 1$，$t\leftarrow 8$，$r\leftarrow 1$开始，之后9—1，$i\leftarrow 2$，$t\leftarrow 6$，$r\leftarrow 3$；接着9—3，9—2，$i\leftarrow 4$，$t\leftarrow 4$，$r\leftarrow 8$；最后9—8。

105. 注意到$d_k\geqslant k$成立当且仅当$c_k\geqslant k$。当$d_k\geqslant k$时，我们有

$$c_1+\cdots+c_k=k^2+\min(k, d_{k+1})+\min(k, d_{k+2})+\cdots+\min(k, d_n);$$

因此条件$d_1+\cdots+d_k \leqslant c_1+\cdots+c_k-k$等价于

$$d_1+\cdots+d_k \leqslant f(k)，此处f(k)=k(k-1)+\min(k, d_{k+1})+\cdots+\min(k,d_n)。 \qquad (*)$$

若$k \geqslant s$我们有$f(k+1)-f(k)=2k-d_{k+1} \geqslant d_{k+1}$。所以(*)式对$1 \leqslant k \leqslant n$均成立当且仅当它对$1 \leqslant k \leqslant s$成立。条件(*)是由P. Erdös和T. Gallai发现的[《Matematikai Lapok》11 (1960), 264—274]。如果我们考虑$\{1, \cdots, k\}$和$\{k+1, \cdots, n\}$之间的边，这显然是必要条件。

设$a_k=d_1+\cdots+d_k-c_1-\cdots-c_k+k$，并假设在步骤H3和H4执行后存在某个$k \leqslant s$使$a_k>0$。记$A_j$, C_j, D_j, N和S为在步骤H3和H4执行之前对应a_j, c_j, d_j, n和s的值。从而$N=n+1$，$D_j = d_j + (0或1)$等。我们想要证明对某些$K \leqslant S$有$A_K>0$。

步骤H3和H4已经去掉了第t列中最下面的q个单元（对某些$t \geqslant S$），以及第$1 \sim p$行的最右侧的单元，其中$q+p=D_N$。从而$A_j=a_j\ (1 \leqslant j \leqslant p)$，更进一步有$A_j=a_j$当$j \geqslant C_t$。

设k是使$a_k>0$成立的最小的k，记$d_k=d$。若$d>t$我们有$k \leqslant p$，于是$A_k=a_k>0$。从而我们可以假设$d=t-(0或1)$且$D_k=t$。

若$k<j \leqslant C_t$我们有$d_j \geqslant D_j-1 = t-1 \geqslant d-1 \geqslant c_k-1 \geqslant c_j-1$。因此当$K=C_t$时，$A_K = a_K \geqslant a_k$；我们可假设$C_t>S$。

现在有$D_S = D_{S+1} = t$，所以$S = t$。也有$k=t$，否则$c_k \geqslant S+1 > t \geqslant d$。从而$s=S$且$d = t = c_t$。进一步的分析表明$A_t \leqslant 0$的唯一可能是$D_j = t+[j \geqslant t]\ (1 \leqslant j \leqslant N=t+2)$。算法H确实把这个“好”序列变成了“坏”序列，然而$D_1+\cdots+D_N=t^2+3t-1$是奇数。

106. 在$d \leqslant 1$且$n \geqslant d+2$的一般情形下是错误的。其余情况正确：事实上，步骤H4中生成的前$n-1$条边不含回路，所以它们形成了一棵生成树。

107. 习题78中的置换φ将度为d的顶点映射到了度为$n-1-d$的顶点。而φ^2是自同构，它将除了可能存在度为$(n-1)/2$的固定点之外的度相同的两点配对。

（相反，算法H的一些复杂扩展可以将满足这些条件的每个graphical序列生成自补图，除非n为奇数时有$d_{(n-1)/2}=(n-1)/2$。详见C. R. J. Clapham和D. J. Kleitman,《J. Combinatorial Theory》B20 (1976), 67—74 。）

108. 我们可以假设$d^+_1 \geqslant \cdots \geqslant d^+_n$，入度$d^-_k$不需要排成特定的顺序。对序列$d_1 \cdots d_n=d^+_1 \cdots d^+_n$运用算法H，但需要做以下改变：步骤H2变为“[是否已完成?]若$d_1 = n = 0$则成功终止，若$d_1 > n$则不成功终止。”在步骤H3设置i、t和r的值之后，如果$d^-_n > c_1$，则不成功终止；否则对$1 \leqslant j \leqslant d^-_n$执行步骤H4，然后设置$n \leftarrow n-1$并回到H2。在步骤H5中，忽略“$c_j \leftarrow c_j-1$”，并添加弧$k \rightarrow n$以代替边$k - n$。类似引理M和推论H的讨论可以确保这个进程。

（7.2.1.4节习题57证明了这样的有向图存在当且仅当$d^-_n+\cdots+d^-_n=d^+_n+\cdots+d^+_n$和$d^-_1 \cdots d^-_n=\{d'_1, \cdots, d'_n\}$，其中$d'_1 \geqslant \cdots \geqslant d'_n$并且$d'_1 \cdots d'_n$被共轭（conjugate）划分$c_1 \cdots c_n=(d^+_1 \cdots d^+_n)^T$所多数化。如果禁止出现自循环$v \rightarrow v$，则更困难；参见D. R. Fulkerson,《Pacific J. Math》10 (1960), 831—836。）

109. 同习题108，设$d^+_k=d_k[k \leqslant m]$且$d^-_k=d_k[k>m]$。

110. 度为$d=d_1$的顶点有p个，度为$d-1$的顶点有q个，其中$p+q=n$。

情况1：$d=2k+1$，当$(u-v) \bmod n \in \{2,3,\cdots,k+1,n-k-1,\cdots,n-3,n-2\}$时，设置$u$—$v$，并添加$p/2$条边1—2, 3—4,⋯,$(p-1)$ — p。

情况2：$d=2k$。当$(u-v) \bmod n \in \{2,3,\cdots,k, n-k,\cdots,n-3,n-2\}$时设置$u$—$v$，并添加边1—2, 3—4,⋯,$(q-1)$—$q$及路径或回路$(q=0?n:q)$—$(q+1)$—⋯—$(n-1)$—$n$。[D. L. Wang和D. J. Kleitman在《Networks》3 (1973), 225—239证明了这样的图是强连通的。]

111. 假设$N=n+n'$且$V'=\{n+1,\cdots,N\}$。我们想要在k和V'之间生成$e_k=d-d_k$条边，并在V'内部附加边，使得V'的任意顶点度为d。设$s=e_1+\cdots+e_n$。这个任务只有在以下条件下是可能的：

(i) $n' \geqslant \max(e_1,\cdots,e_n)$；(ii) $n'd>s$；(iii)$n'd<s+n'(n-1)$；(iv) $(n+n')d$ 是偶数。

这样的边当n'满足(i)～(iv)时是存在的：首先，由(i)，V和V'之间s条符合要求的边可通过循环选择端点$(n+1, n+2, \cdots, n+n', n+1, \cdots)$而生成。这个过程给每个$V'$中的顶点分配了$\lfloor s/n' \rfloor$或$\lceil s/n' \rceil$条边。由(ii)我们有$\lceil s/n' \rceil \leqslant d$，由(iii)有$d-\lfloor s/n' \rfloor \leqslant n'-1$。因而在$V'$内部添加的边由习题110和(iv)是可构造的。

关于$n'=n$的选择总是有效的。相反，若$G = K_n(V) \setminus \{1—2\}$，条件(iii)要求$n' \geqslant n$，当$n \geqslant 4$时。[P. Erdös和P. Kelly，《AMM》70 (1963)，1074—1075。]

112. *Miles*数据中唯一的最佳三角形是`Saint Louis,MO`$\overset{748}{—}$`Toronto,ON` $\overset{746}{—}$ `Winston-Salem, NC`$\overset{748}{—}$`SaintLouis, MO`。

113. 由Murphy法则，它有n行m列，因此是$n \times m$，而不是$m \times n$。

114. 多重图中的自循环是有重复端点的边$\{a, a\}$，并且多重图是2—致（2-uniform）超图。所以当一条边含一个顶点超过一次时，我们允许一般超图的关联矩阵有大于1的元素。（比较讲究的人可能将其称为多重超图（multihypergraph））对应(26)的关联矩阵和二分图分别是

$$\begin{pmatrix} 210000 \\ 011100 \\ 001122 \end{pmatrix};$$



115. B^TB的第e行、第f列的元素是$\sum_v b_{ve}b_{vf}$，故B^TB是$2I$加上$L(G)$的邻接矩阵。类似的BB^T是D加上G的邻接矩阵，其中D是对角矩阵，有d_{vv}=v的度。（参见2.3.4.2节的习题18、19和20。）

116. $\overline{K_{m,n}^{(r)}} = K_m^{(r)} \oplus K_n^{(r)}$，将(38)推广至对所有的$r \geqslant 1$。

117. M=4及$V=\{0,1,2\}$时，具有单边（singleton edge）但互不同构的多重集是$\{\{0\}, \{0\}, \{0\}, \{0\}\}$, $\{\{0\}, \{0\}, \{0\}, \{1\}\}$, $\{\{0\}, \{0\}, \{1\}, \{1\}\}$和$\{\{0\}, \{0\}, \{1\}, \{2\}\}$。

答案一般是将m分成最多n部分的拆分数，即$\left|\begin{matrix} m+n \\ n \end{matrix}\right|$，使用7.2.1.4节中的记法。（当然，没有理由去考虑1—致超图的划分，除非回答奇怪的问题。）

118. 设d是顶点度的总和，相应二分图有$m+n$个顶点，d条边，p个成分。所以由定理2.3.4.1A得$d = m+n-p$。

119. 这样就有另外一条边，包含了所有7个顶点。

120. 我们能断言（超）弧是任意的顶点序列或不同顶点的序列。但大多作者似乎定义超弧为$A\to v$，其中A是无序的顶点集。最好的定义可能是有最多实际应用的那个。

121. $\chi(H) = |F| - \alpha(I(H)^T)$是$V$用集合$F$最小覆盖的规模。

122. (a)我们可以证明只有7个3元素覆盖，即一条边的顶点；从而有7个4元素独立集，即一条边的补。我们不能对超图进行2着色，因为一种颜色需要用到4次而另3种颜色成为一条边。(超图(56)是7点7线的投影平面。)

(b)因为我们在对偶化（dualize），让我们将Petersen图的顶点和边称为“点”和“线”。于是对偶的顶点和边分别是线和点。将外点与内点相连的5条线着成红色，另外10条线是独立的（它们不包含那3条接触任何点的线），因此它们可以被着成绿色。没有11条线的集合可以是独立的，因为没有4条线可以接触所有10个点。(因此Petersen图的对偶是二分超图，尽管它有长为5的回路。)

123. 他们对应于$n\times n$拉丁方阵，它的元素为顶点的颜色。

124. 4种颜色很容易满足。若它可以被3着色，每种颜色必然着4个点，因为没有5个顶点是独立的。从而2个相反的角必然着相同颜色，推出矛盾。

125. Chváta图是此类最小的g=4的图。G. Brinkmann找到了g=5时的最小图：共有21个顶点a_j, b_j, c_j $(0\leqslant j<7)$，边为$a_j — a_{j+2}$, $a_j — b_j$, $a_j — b_{j+1}$, $b_j — c_j$, $b_j — c_{j+2}$, $c_j — c_{j+3}$，下标是mod 7的。M. Meringer验证了若$g>5$，至少有35个点。B. Grünbaum猜想g可以任意大，但是没有进一步的结构是已知的。[参见《AMM》77 (1970)，1088—1092；《Graph Theory Notes of New York》32 (1997), 40—41。]

126. 当m和n为偶数，并且C_m和C_n均为二分图时，易被4着色。其他情形4着色是不可能的。当$m=n=3$时，由习题93知9着色是最优的。当$m=3$，$n=4$或5时，至多两个顶点是独立的；容易找到最优的6或8着色。其余情况可以得到5着色，若将顶点(j, k)着色(a_j+2b_k) mod 5，其中循环序列$\langle a_j\rangle$和$\langle b_k\rangle$的循环周期分别是m和n，并且$a_j-a_{j+1}\equiv\pm1$且$b_k-b_{k+1}\equiv\pm1$，对所有的j和k。[K. Vesztergombi,《Acta Cybernetica》4 (1978), 207—212。]

127. (a)当n=1时结果是正确的。否则，设$H=G\backslash v$，其中v是任一顶点，则$\overline{H}=\overline{G}\backslash v$，并由递归可知$\chi(H)+\chi(\overline{H})\leqslant n$。易知$\chi(G)\leqslant\chi(H)+1$及$\chi(\overline{G})\leqslant\chi(\overline{H})+1$，所以不会出现问题，除非等号在三种情况中成立，但这是不可能的。于是有$\chi(H)\leqslant d$和$\chi(\overline{H})\leqslant n-1-d$，其中$d$是$G$中$v$的度。[E. A. Nordhaus和J. W. Gaddum,《AMM》63 (1956), 175—177。]

为得到等号成立条件，设$G=K_a\oplus\overline{K_b}$，其中$ab>0$，$a+b=n$。那么我们有$\overline{G}=\overline{K_a}-K_b$，$\chi(G)=a$，且$\chi(\overline{G})=b+1$。[所有使等号成立的图已被H.-J. Finck找到，参见《Wiss. Zeit. der Tech. Hochschule Ilmenau》12 (1966), 243—246。]

(b) G的一个k着色中至少有$\lceil n/k\rceil$个顶点是相同颜色的，这些顶点形成了$\overline{G}$中的一个团。从而$\chi(G)\chi(\overline{G})\geqslant\chi(G)\lceil n/\chi(G)\rceil\geqslant n$，当$G=K_n$时等号成立。

(从(a)和(b)我们可以推断$\chi(G)+\chi(\bar{G})\geqslant 2\sqrt{n}$和$\chi(G)\chi(\bar{G})\leqslant \frac{1}{4}(n+1)^2$。)

128. $\chi(G\square H)=\max(\chi(G), \chi(H))$。这么多颜色很明显是必要的。若函数$a(u)$和$b(v)$用颜色$\{0, 1, \cdots, k-1\}$着色$G$和$H$，那么我们可以用$c(u, v) = (a(u) + b(v)) \bmod k$着色$G\square H$。

129. 一个完整的行或列（16种情况）；一个完整的长度大于等于4的对角线（18种情况）；一个5单元的模式$\{(x, y), (x-a, y-a), (x-a, y+a), (x+a, y-a), (x+a, y+a)\}$，其中$a\in\{1, 2,3\}$（36+16+4种情况）；一个5单元的模式$\{(x,y), (x-a, y), (x+a, y), (x, y-a), (x, y+a)\}$，其中$a\in\{1,2,3\}$（36+16+4种情况）；一个模式，当第5个单元在棋盘外，该模式包含了那5个单元中的4个（24+32+24种情况）；或者一个4单元的模式 $\{(x,y), (x+a, y), (x, y+a), (x+a, y+a)\}$对于$a\in\{1, 3, 5, 7\}$(49+25+9+1种情况)。

总共310个最大团，对应规模(4, 5, 6, 7,8)分别为(168, 116, 4, 4,18)。

130. 若图G有p个最大团，图H有q个，那么$G—H$有pq个，因为$G—H$的团是G和H团的简单并。更进一步，空图$\overline{K_n}$有n个最大团，即它是单点集（singleton）。

于是各部分规模为$\{n_1, \cdots, n_k\}$的完全k分图是相应规模的空图的结合，有$n_1\cdots n_k$个最大团。

131. 假设$n>1$。在一个完全k分图中，当每部分规模为3(除了可能1个或2个部分的规模为2)，$n_1\cdots n_k$已被最大化。(参见7.2.1.4节的习题68(a)。)我们需要证明在任意图中$N(n)$不能比它再大了。

设$m(v)$为包含顶点v的最大团的个数。若$u \not- v$且$m(u) \leqslant m(v)$，构造图G'类似G，除了u现在邻接的是v的所有邻居，而不是它先前的邻居。任一图中每个最大团U必然是以下三类中的一种：

i) $u\in U$；G中这样的有$m(u)$个，G'中有$m(v)$个。

ii) $v\in U$；G中这样的有$m(v)$个，G'中也是。

iii) $u\notin U$且$v\notin U$；G中这样的最大团在G'中也是最大团。

因此G'的最大团数量至少和G的一样多，我们可以通过适当地重复这个步骤来得到完全k分图。[这个论证，归因于Paul Erdös，由J. W. Moon和L. Moser 在《Israel J. Math》3 (1965), 23—25中给出。]

132. G和H的团的强积（strong product）是$G\boxtimes H$中的团（由习题93知）。从而有$\omega(G\boxtimes H)\geqslant\omega(G)\omega(H) = \chi(G)\chi(H)$，另一方面，由$G$和$H$的着色$a(u)$和$b(v)$导出$G\boxtimes H$的着色$c(u, v) = (a(u),b(v))$，故$\chi(G\boxtimes H)\leqslant\chi(G)\chi(H)$，并且$\omega(G\boxtimes H)\leqslant\chi(G\boxtimes H)$。

133. (a) 24; (b) 60; (c) 3; (d) 6; (e) 6; (f) 4; (g) 5; (h) 4; (i) $K_2\boxtimes C_{12}$; (j) 18; (k) 12。(l)是的，度为5。(m)不是。[能画出少于12个交叉点的图吗？](n)是的，事实上，它是4连通的（参见7.4.1节）(o)是的，我们认为每个图都是有向图，每条边带两个箭头。(p)显然不是。(q)易知，是的。

[音乐图代表调号（key signature）之间的简单转调。它第一次出现在R. J. Wilson和J. J. Watkins (1990)的《Graphs》一书第73页。]

134. 通过转动或互换内外顶点，我们可以找到把所有顶点映到C的自同构。若C固定，我们可以交换剩下的11对内外顶点中的任意子集，并且/或者做一次左右反射。因而共有$24\times 2^{11}\times 2$=98 304个自同构。

135. 设$\omega=e^{2\pi i/6}$，并定义矩阵$Q=(q_{ij})$, $S=(s_{ij})$，其中$q_{ij}=[j=(i+1) \bmod 12]$和$s_{ij}=\omega^{ij}$ $(0\leqslant i,j<12)$由习题96(b)，音乐图$K_2 \boxtimes C_{12}$的邻接矩阵为$A=\begin{pmatrix}1&1\\1&1\end{pmatrix}\otimes(I+Q+Q^-)-I$。设$T$为矩阵$\begin{pmatrix}1&1\\1&-1\end{pmatrix}\otimes S$，则$T^-AT$是一个对角矩阵$D$，其前12个元素为$1+4\cos\frac{j\pi}{6}$ $(0\leqslant j<12)$，另12个元素为−1。故$A^{2m}=TD^mT^-$，从C到$(C, G, D, A, E, B, F^\#)$的2m步路的数量相应为

$$C_m=\frac{1}{24}(25^m+2(13+4\sqrt{3})^m+3^{2m+1}+2(13-4\sqrt{3})^m+16)$$

$$G_m=\frac{1}{24}(25^m+\sqrt{3}(13+4\sqrt{3})^m-\sqrt{3}(13-4\sqrt{3})^m-1)$$

$$D_m=\frac{1}{24}(25^m+(13+4\sqrt{3})^m+(13-4\sqrt{3})^m-3)$$

$$A_m=\frac{1}{24}(25^m-3^{2m+1}+2)$$

$$E_m=\frac{1}{24}(25^m-(13+4\sqrt{3})^m-(13-4\sqrt{3})^m+1)$$

$$B_m=\frac{1}{24}(25^m-\sqrt{3}(13+4\sqrt{3})^m+\sqrt{3}(13-4\sqrt{3})^m-1)$$

$$F^\#_m=\frac{1}{24}(25^m-2(13+4\sqrt{3})^m+3^{2m+1}-2(13-4\sqrt{3})^m)$$

另外有$a_m=C_m-1$, $d_m=F_m=e_m=G_m$等。特别地，$(C_6, G_6, D_6, A_6, E_6, B_6, F^\#_6)$=(15462617, 14689116, 12784356, 10106096, 7560696, 5655936, 5015296)。所以所求概率为$15462617/5^{12}\approx 6.33\%$。当$m\to\infty$，概率为$\frac{1}{24}+O(0.8^m)$。

136. 不。阶为10的Cayley图中只有两个是立方体，即$K_2\square C_5$（它们的顶点可以记为$\{e, \alpha, \alpha^2, \alpha^3, \alpha^4, \beta, \beta\alpha, \beta\alpha^2, \beta\alpha^3, \beta\alpha^4\}$，其中$\alpha^5=\beta^2=(\alpha\beta)^2=e$)，并且该图的顶点集为{0,1, ⋯,9}，有向边为$v\to(v\pm 1) \bmod 10$，$v\to(v+5) \bmod 10$。[参见D. A. Holton 和 J. Sheehan,《The Petersen Graph》(1993), 习题 9.10。顺便说一句，SGB图 *raman* $(p, q, t, 0)$是Cayley图。]

137. 记$[x, y]$为(x, y)的标号（label），我们希望对所有的x, y有$[x, y]=[x+a, y+b]=[x+c, y+d]$。若$\mathbf{A}$是矩阵$\begin{pmatrix}a&b\\c&d\end{pmatrix}$，$\mathbf{A}$的最底行加到$\mathbf{A}$的最顶行这样的变换进行$t$次后，$\mathbf{A}$变为矩阵$\mathbf{A}'=\begin{pmatrix}1&t\\0&1\end{pmatrix}\mathbf{A}=\begin{pmatrix}a'&b'\\c'&d'\end{pmatrix}$，其中$a'=a+tc$，$b'=b+td$，$c'=c$，$d'=d$。新条件[$x$,

$y]=[x+a', y+b']=[x+c', y+d']$等价于老条件，并有$\gcd(a', b', c', d')=\gcd(a, b, c, d)$。类似地，我们可以左乘$\begin{pmatrix}1 & 0\\ t & 1\end{pmatrix}$而不改变原问题。

我们也可以对列进行变换，用$\boldsymbol{A}''=\boldsymbol{A}\begin{pmatrix}1 & t\\ 0 & 1\end{pmatrix}=\begin{pmatrix}a'' & b''\\ c'' & d''\end{pmatrix}$代替$\boldsymbol{A}$，其中$a''=a$, $b''=ta+b$, $c''=c$, $d''=tc+d$。这个操作改变了原问题，但仅仅是微小的改动：若我们能找到标号满足对任意的x，y有$[[x, y]]=[[x+a'', y+b'']]=[[x+c'', y+d'']]$，那么我们有$[x, y]=[x+a, y+b]=[x+c, x+d]$如果$[x, y]=[[x, y+tx]]$。类似地，我们可以将$\boldsymbol{A}$右乘$\begin{pmatrix}1 & 0\\ t & 1\end{pmatrix}$，问题几乎不变。

一系列这样的行列变换会把$\boldsymbol{A}$变为简单的形式，$\boldsymbol{UAV}=\begin{pmatrix}1 & 0\\ 0 & n\end{pmatrix}$，其中$\boldsymbol{U}$和$\boldsymbol{V}$是满足$\det\boldsymbol{U}=\det\boldsymbol{V}=1$的整数矩阵。若我们还有$\boldsymbol{V}=\begin{pmatrix}\alpha & \beta\\ \gamma & \delta\end{pmatrix}$，定义$[x, y]=[[\alpha x+\gamma y, \beta x+\delta y]]$，满足简单条件$[[x, y]]=[[x+1, y]]=[[x, y+n]]$的转化问题提供了原标号问题的一个解。

最后转化后的标号问题较简单：我们设$[[x, y]]=y \bmod n$则所需答案是设置$p=\beta$, $q=\delta$。

138. 换成$k\times k$矩阵$\boldsymbol{A}$，其余同前，行列变换将问题转化成对角矩阵$\boldsymbol{UAV}$。对角线元素$(d_1, \cdots, d_k)$有如下特征：$d_1\cdots d_j$为$\boldsymbol{A}$的所有$j\times j$的子矩阵的行列式值的最大公因子。[此即“Smith范式”，参见H. J. S. Smith,《Philosophical Transactions》151 (1861), 293—326, §14。] 若标号$[[x]]$满足转化后的问题，原问题满足$[x]=[[x\boldsymbol{V}]]$。在广义圆环面上的元素个数为$n=\det\boldsymbol{A}=d_1\cdots d_k$。

若$d_1=\cdots=d_{k-1}=1$，转化后的问题仍像原问题那样有简单解。但是更一般地转化后的标号是维$(d_{k-r+1},\cdots, d_k)$的一个r维的普通圆环面，此处$d_{k-r+1}>d_{k-r}=1$。（这里$d_0=1$，可能有$r=k$）

在所求例子中，我们有$d_1=1, d_2=2, d_3=10, n=20$；事实上，

$$\boldsymbol{UAV}=\begin{pmatrix}1 & -2 & 0\\ 0 & 1 & -1\\ -1 & -1 & 4\end{pmatrix}\begin{pmatrix}3 & 1 & 1\\ 1 & 3 & 1\\ 1 & 1 & 3\end{pmatrix}\begin{pmatrix}1 & 5 & 6\\ 0 & 1 & 1\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 10\end{pmatrix}$$

每一个点(x, y, z)得到一个2维标号$(u, v)=((5x+y) \bmod 2, (6x+y+z) \bmod 10)$。$(u, v)$的6个邻居是$((u\pm 1) \bmod 2, v)$, $((u\pm 1) \bmod 2, (v\pm 1) \bmod 10)$, $(u, (v\pm 1) \bmod 10)$。这是一个多重图，因为前两个邻居是相等的。但又不是度为8的多重图$C_2 \boxtimes C_{10}$。

[广义圆环面（generalized toruses）必然是阿贝尔群的Cayley图，参见习题136。它们已被作为网络互联的方法提出，在应用中k和n固定，而半径最小。参见C. K. Wong和D. Coppersmith,《JACM》21 (1974), 392—402; C. M. Fiduccia、R. W. Forcade和J. S. Zito,《SIAM J. Discrete Math》11(1998), 157—167。]

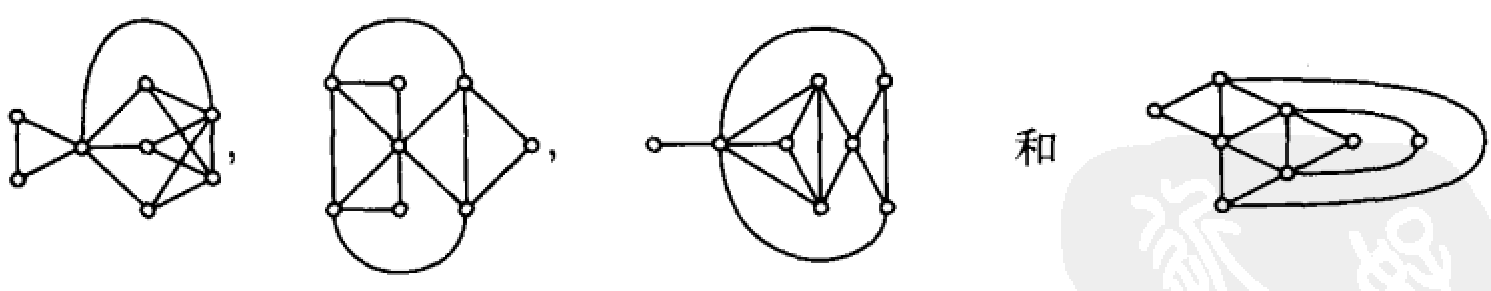
139. (本习题帮助区分标号图G和未标号的图H，前者顶点有确定名称，而后者如图2中的那些)若N_H是$\{1, 2, \cdots, h\}$上同构于H的标号图的数量，U是V的任意h元子集，则$G|H$同构于H的概率是$N_H/2^{h(h-1)/2}$，因此答案是$\binom{n}{h}N_H/2^{h(h-1)/2}$。我们仅需计算出$N_H$的值，即(a)1；(b)$h!/2$；(c)$(h-1)!/2$; (d) $h!/a$。其中H有一个自同构。

140. (a)$\#(K_3{:}W_n)=n-1$且$\#(P_3{:}W_n)=\binom{n-1}{2}$ $n\geqslant 5$；也有$\#(\overline{K_3}{:}W_8)=7$。

(b)G是成比例的当且仅当$\#(K_3{:}G)=\#(\overline{K_3}{:}G)=\frac{1}{8}\binom{n}{3}$且$\#(P_3{:}G)=\#(\overline{P_3}{:}G)=\frac{3}{8}\binom{n}{3}$。若$G$有$e$条边，我们有$(n-2)e=3\#(K_3{:}G)+2\#(P_3{:}G)+\#(\overline{P_3}{:}G)$，因为每对顶点出现在$n-2$个导出子图中。若$G$有度序列$d_1\cdots d_n$，则$d_1+\cdots+d_n=2e$，$\binom{d_1}{2}+\cdots+\binom{d_n}{2}=3\#(K_3{:}G)+\#(P_3{:}G)$，并且$d_1(n-1-d_1)+\cdots+d_n(n-1-d_n)=2\#(P_3{:}G)+2\#(\overline{P_3}{:}G)$。因此成比例的图满足(*)——除非$n=2$（这道习题不包含这种情况。）

相反地，若G满足(*)并有合理的$\#(K_3{:}G)$，则它也有合理的$\#(P_3{:}G)$，$\#(\overline{P_3}{:}G)$和$\#(\overline{K_3}{:}G)$。[参考：S. Janson和J. Kratochvíl，《Random Structure & Algorithms》2 (1991)，209—224。在《J. Combinatorial Theory》B47 (1989), 125—145中，A. D. Barbour、M. Karoński 和A. Ruciński 证明了变量$\#(H{:}G)$与n^{2h-2}、n^{2h-3}或n^{2h-4}成比例，其中第一种情况当H没有$\frac{1}{2}\binom{n}{2}$条边时出现，而第三种情况当H是成比例的图时出现。]

141. 只有8个度序列$d_1\cdots d_8$满足(*)：73333333(1/2), 65433322(26/64), 64444222(2/10), 64443331(8/22), 55543222(8/20) ,55533331(2/10), 55444321(26/64)和44444440(1/2)。每个度序列在下方列出，(N_1/N)满足N个互不同构的图有这个序列并且它们中N_1个是成比例的。后3种情况是前3种的补。没有秩为8的图既是成比例的又是自补的。前5种情况中最大对称的例子为W_8。



142. 提示同前面的答案。$(n-3)\#(\overline{K_3}{:}G)$和$(n-3)\#(P_3{:}G)$也可以用4顶点计数表示。更进一步，有$e$条边的图有$\binom{e}{2}=\#(P3\subseteq G)+\#(K_2\oplus K_2\subseteq G)$，因为任意两条边都形成$P_3$或$K_2\oplus K_2$；在这个公式中，$\#(P_3\subseteq G)$是非必要导出的子图的数目。

我们有$\#(P_3\subseteq G)=\#(P_3{:}G)+3\#(K_3{:}G)$，而类似的公式用诱导计数表达了

$\#(K_2 \oplus K_2 \subseteq G)$。所以一个超成比例图（extraproportional graph）必然是成比例的并且满足$e=\frac{1}{2}\binom{n}{2}$，$\#(P3 \subseteq G)=\frac{3}{4}\binom{n}{3}$，$\#(K_2 \oplus K_2 \subseteq G)=\frac{3}{4}\binom{n}{4}$。但是这些值与$\binom{e}{2}$的公式矛盾。

143. 考虑这样的图：它的顶点为A的行，它的边u—v意味着行u和v除了第j列外是相同的。将这样的边标记为j。

若图中包含了一个回路，删除这个回路上的任意一条边，重复这个过程直到不再有回路为止。注意到每条删除的边的标号在这个回路的其他地方出现，所以这样删除不影响边标号的集合。但是由定理2.3.4.1A知，剩下少于$m \leqslant n$条边；所以不同的标号少于n种。[参见 J. A. Bondy，《J. Combinatorial Theory》B12 (1972), 201—202。]

144. 设G是顶点集为$\{1,\cdots,m\}$，边为i—j的图，当且仅当$* \neq x_{il} \neq x_{jl} \neq *$，对某些$l$。这个图是$k$可着色的当且仅当存在至多$k$个不同的行的一个完全图。反过来，若图$G$有顶点$\{1,\cdots,n\}$、邻接矩阵$A$，则$n \times n$矩阵$X=A+*(J-I-A)$有性质$i$—$j$当且仅当$* \neq x_{il} \neq x_{jl} \neq *$，对某些$l$。[参见M. Sauerhoff和I. Wegener，《IEEE Trans》CAD-15 (1996)，1435—1437。]

145. 设置$c \leftarrow 0$并对$1 \leqslant j \leqslant n$重复以下操作：若$c=0$，设置$x \leftarrow a_j$及$c \leftarrow 1$；若$x=a_j$，设置$c \leftarrow c+1$；否则设置$c \leftarrow c-1$。最后$x$就是所求答案。思想是保持最主要元素$x$的踪迹，它在没去掉的元素中出现了$c$次；当找到$x \neq a_j$时，我们去掉$a_j$和一个$x$。[参见《Automated Reasoning》(Kluwer, 1991), 105—117。扩展问题：在$O(n \log k)$步内找出所有出现超过n/k次的元素，J. Misra和D. Gries在《Science of Computer Programming》2 (1982)，143—152中讨论过。]

7.1.1节

1. （C. Sartena给出的解答）他曾经一直在描述$x \Rightarrow y$的含义，其中“它”分别代表y, x, x, y, y, x。（其他解法也可以。）

2. 对应于平卡斯人的$x \circ y$，地球上的运算是$\overline{\bar{x} \circ \bar{y}}$；因此它的真值表是（地球上的）$\circ$的真值表的补的反转。故答案分别是$\top, \vee, \subset, \mathsf{L}, \supset, \mathsf{R}, \equiv, \wedge, \bar\wedge, \oplus, \bar{\mathsf{R}}, \bar\supset, \bar{\mathsf{L}}, \bar\subset, \bar\vee, \perp$（任何涉及表1中16种运算的恒等式都有着对偶恒等式，它们可通过上述平卡斯变换而得。比如，德摩根定律（11）和（12）是互对偶的，同样，恒等式（3）和（4）则与$\equiv$和$\oplus$相联系。据此可以认为$\equiv$与它的对偶$\oplus$一样有用）。

3. (a) $\vee$； (b) $\wedge$； (c) $\bar{\mathsf{L}}$； (d) $\equiv$。[实际上如果用-1代表真而用$+1$代表假，许多公式的结果会更有意义，即使这种约定似乎有一点点不合规则；此时$x \cdot y$对应$\oplus$。注意在这两种约定下都有$\langle xyz \rangle = \mathrm{sign}(x+y+z)$。]

4. [《Trans.Amer.Math.Soc》14 (1913, 481—488)] (a)从L的R的真值表开始；然后从每个已知的真值表对α和β，按位计算$\alpha \bar\wedge \beta$的真值表，按公式长度顺序生成结果并写下通向新的4位真值表的最短公式：

$\perp$: $(x \bar\wedge (x \bar\wedge x)) \bar\wedge (x \bar\wedge (x \bar\wedge x))$
$\wedge$: $(x \bar\wedge y) \bar\wedge (x \bar\wedge y)$
$\bar\supset$: $(x \bar\wedge (x \bar\wedge y)) \bar\wedge (x \bar\wedge (x \bar\wedge y))$
L: x
$\bar\subset$: $(y \bar\wedge (x \bar\wedge x)) \bar\wedge (y \bar\wedge (x \bar\wedge x))$
R: y
$\oplus$: $(y \bar\wedge (x \bar\wedge x)) \bar\wedge (x \bar\wedge (x \bar\wedge y))$
$\vee$: $(y \bar\wedge y) \bar\wedge (x \bar\wedge x)$
$\bar\vee$: $(x \bar\wedge (x \bar\wedge x)) \bar\wedge ((y \bar\wedge y) \bar\wedge (x \bar\wedge x))$
$\equiv$: $(x \bar\wedge y) \bar\wedge ((y \bar\wedge y) \bar\wedge (x \bar\wedge x))$
$\bar{\mathsf{R}}$: $y \bar\wedge y$
$\subset$: $y \bar\wedge (x \bar\wedge x)$
$\bar{\mathsf{L}}$: $x \bar\wedge x$
$\supset$: $x \bar\wedge (x \bar\wedge y)$
$\bar\wedge$: $x \bar\wedge y$
$\top$: $x \bar\wedge (x \bar\wedge x)$

(b) 在这种情况下，从对应于$\perp$、$\top$、L、R的4个真值表开始，当有机会对给定长度的公式进行选择时，选取含较少变量的公式：

$\perp$: 0
$\wedge$: $(x \bar\wedge y) \bar\wedge 1$
$\bar\supset$: $((y \bar\wedge 1) \bar\wedge x) \bar\wedge 1$
L: x
$\bar\subset$: $(y \bar\wedge (x \bar\wedge 1)) \bar\wedge 1$
R: y
$\oplus$: $(y \bar\wedge (x \bar\wedge 1)) \bar\wedge ((y \bar\wedge 1) \bar\wedge x)$
$\vee$: $(y \bar\wedge 1) \bar\wedge (x \bar\wedge 1)$
$\bar\vee$: $1 \bar\wedge ((y \bar\wedge 1) \bar\wedge (x \bar\wedge 1))$
$\equiv$: $(x \bar\wedge y) \bar\wedge ((y \bar\wedge 1) \bar\wedge (x \bar\wedge 1))$
$\bar{\mathsf{R}}$: $y \bar\wedge 1$
$\subset$: $y \bar\wedge (x \bar\wedge 1)$
$\bar{\mathsf{L}}$: $x \bar\wedge 1$
$\supset$: $(y \bar\wedge 1) \bar\wedge x$
$\bar\wedge$: $x \bar\wedge y$
$\top$: 1

5. (a) $\perp$: $x \bar\subset x$； $\wedge$: $(x \bar\subset y) \bar\subset y$； $\bar\supset$: $y \bar\subset x$； L: x； $\bar\subset$: $x \bar\subset y$； R: y；其他10个无法表示。(b)然而加上常量，所有16个都可以表示：

$\perp$: 0
$\wedge$: $(y \bar\subset 1) \bar\subset x$
$\bar\supset$: $y \bar\subset x$
L: x
$\bar\subset$: $x \bar\subset y$
R: y
$\oplus$: $((y \bar\subset x) \bar\subset ((x \bar\subset y) \bar\subset 1)) \bar\subset 1$
$\vee$: $(y \bar\subset (x \bar\subset 1)) \bar\subset 1$
$\bar\vee$: $y \bar\subset (x \bar\subset 1)$
$\equiv$: $(y \bar\subset x) \bar\subset ((x \bar\subset y) \bar\subset 1)$
$\bar{\mathsf{R}}$: $y \bar\subset 1$
$\subset$: $(x \bar\subset y) \bar\subset 1$
$\bar{\mathsf{L}}$: $x \bar\subset 1$
$\supset$: $(y \bar\subset x) \bar\subset 1$
$\bar\wedge$: $((y \bar\subset 1) \bar\subset x) \bar\subset 1$
$\top$: 1

[B. A. Bernstein, 《University of California Publications in Mathematics》 1 (1914), 87—96]

6. (a) $\bot, \wedge, \mathrm{L}, \mathrm{R}, \oplus, \vee, \equiv, \top$。(b) $\bot, \mathrm{L}, \mathrm{R}, \oplus, \equiv, \top$。[注意，所有操作都是可结合的。实际上，所述的恒等式蕴涵了一般的结合律：首先我们有(i) $(x\circ y)\circ((z\circ y)\circ w)=((x\circ z)\circ(z\circ y))\circ((z\circ y)\circ w)=(x\circ z)\circ w$，类似地，我们有(ii)$(x\circ(y\circ z))\circ(y\circ w)= x\circ(z\circ w)$。进一步由(i)有(iii)有$(x\circ y)\circ(z\circ w)=(x\circ y)\circ((z\circ y)\circ(y\circ w))=(x\circ z)\circ(y\circ w)$。这样由 (i), (ii), (iii)有$(x\circ z)\circ w=(x\circ z)\circ((z\circ z)\circ w)=(x\circ(z\circ z))\circ(z\circ w)= x\circ(z\circ w)$。由$\{x_1,\cdots,x_n\}$产生的自由系统恰好有$n+2^n n^2$个不同的元素，即$\{x_j|1\leqslant j\leqslant n\}$和$\{x_i\circ x_{j1}\circ\cdots\circ x_{j_r}\circ x_k|r\geqslant 0$且$1\leqslant i, k\leqslant n$且$1\leqslant j_1<\cdots j_r\leqslant n\}$。]

7. 由等价性，我们希望有恒等式$y\circ(x\circ y)=x$，它只对$\oplus$和$\equiv$成立。[Jevons在《Pure Logic》一书的第151节描述了$\oplus$的这个性质，但他并没有继续这方面的研究。7.2.3节将研究具有该性质的被称为“摸索”（gropes）的一般系统。]

8. （$\{\bot,\wedge,\overline{\subset}\}$, S_0)，（$\{\top,\vee,\supset\}$, S_1)，（$\{\mathrm{L},\bar{\mathrm{L}}\}$, $S_0\cap S_1$)，（$\{\oplus,\equiv,\bar{\mathrm{R}}\}$, S_2)，（$\{\overline{\supset},\overline{\vee}\}$, $S_0\cap S_2$)，（$\{\subset,\overline{\wedge}\}S_1\cap S_2$))和($\mathrm{R}$, any)，其中$S_0=\{\square|0\square 0=0\}$, $S_1=\{\square|1\square 1=1\}$，$S_2=\{\square|\bar{x}\square\bar{y}=\overline{x\square y}\}=\{\mathrm{L},\mathrm{R},\bar{\mathrm{L}},\bar{\mathrm{R}}\}$。这样256对中有92对是满足左分配律的。[在E. Schröder去世后出版的书籍《Vorlesungen über die Algebra der Logik》2, 2(1905)第55节最早讨论了这个问题和习题6。他认为从本质上说，$(\circ,\square)$的真值表$(pqrs, wxyz)$必须满足关系 $((pq\vee rs)\wedge\bar{z})\vee((\bar{p}\bar{q}\vee\bar{r}\bar{s})\wedge w)\vee((p\bar{q}\vee r\bar{s})\wedge((w\equiv z)\vee(x\equiv y)))=0$，由此他给出了问题的答案。]

9. (a)错误，$(x\oplus y)\vee z=(x\vee z)\oplus(y\vee z)\oplus z$；(b)正确，因为当$z=0$和$z=1$时恒等式显然成立；(c)正确，亦即$(x\oplus y)\vee(x\oplus z)=1-[x=y=z]$。

10. 分解（16）的第一阶段产生了真值表为$g=10100011$和$h=10100011\oplus 10010011=00110000$的函数；并且这个过程以类似的方式继续，最后产生$1+y+xz+w+wy+wx+wxz$（对2取模）。

11. 所述的项出现，当且仅当$x_1=x_4=x_5=x_7=x_9=x_{10}=\cdots=0$时$f(x_1,\cdots,x_n)$奇数次为真(在除了$k$个变量外的所有变量均被设置为0后，有$2^k$种情形要考虑)。换句话说，多重线性可以使用下面建议的记法表示，如

$$f(x,y,z)=(f_{000}=f_{00*}z+f_{0*0}y+f_{0**}yz+f_{*00}x+f_{*0*}xz+f_{**0}xy+f_{***}xyz)=\bmod 2$$

这里举例说明的是$n=3$时的情况，其中$f_{**0}=f(1,1,9)\oplus f(1,0,0)\oplus f(0,1,0)\oplus f(0,0,0)$等。

12. (a) 在(23)里用$\bar{w}$代替$1-w$等，可得$1-y-xz+2xyz-w+wy+wx+wxz-2wxyz$。[一些作者曾称这个表达式为“Zhegalkin多项式”，但I. I. Zhegalkin在处理表达式时总是进行模2计算的。文献中出现的其他名字有“可用多项式”、“可靠多项式”、“特征多项式”。]

(b)任意n元函数对应的系数的绝对值可以大到2^{n-1}（通过归纳法可证这是最大的值）。比如，$x_1\oplus\cdots\oplus x_n$的整系数多重线性表示为$e_1-2e_2+4e_3-\cdots+(-2)^{n-1}e_n$，其中$e_k$为$\{x_1,\cdots,x_n\}$的第$k$个初等对称函数。前面答案里的公式在整系数上变成

$$f(x,y,z)=f_{000}+f_{00*}z+f_{0*0}y+f_{0**}yz+f_{*00}x+f_{*0*}xz+f_{**0}xy+f_{***}xyz$$

其中$f_{**0}=f(1,1,0)-f(1,0,0)-\mathrm{f}(0,1,0)+f(0,0,0)$等。这个展开式是Hadamard变换的变相形式，4.6.4节的等式(38)。

(c, d)多项式是如$x_1(1-x_2)(1-x_3)x_4$这样的极小项的和。每个极小项对于$0\leqslant x_1,\cdots x_n\leqslant 1$是非负的，而且所有极小项的和为1。

(e) $\partial f/\partial x_j=h(x)-g(x)$，其中由(d)有$h(x)\geqslant g(x)$（参见习题21）。

13. 实际上，F恰好是整系数多重线性表达式（参见习题12）。

14. 令$r_j=p_j/(1-p_j)$。想要$f(0,0,0)=0$且$f(1,1,1)=1\Leftrightarrow r_1r_2r_3>1$，$f(0,0,1)=0$且$f(1,1,0)=1\Leftrightarrow r_1r_2>r_3$, $f(0,1,0)=0$且$f(1,0,1)=1\Leftrightarrow r_1r_3>r_2$, $f(0,1,1)=0$且$f(1,0,0)=1\Leftrightarrow r_1>r_2r_3$。所以得到(a) $\langle x_1x_2x_3\rangle$；(b) x_1；(c) $\bar{x}_3$。

15. 1.2.6节习题10告诉我们$\binom{x}{k}\bmod 2=[x\&k=k]$。因此，比如，当$x=(x_n\cdots x_1)_2$时，我们有$\binom{x}{11}\equiv x_4\wedge x_2\wedge x_1(\text{modulo }2)$，而且我们能以这种方式得到如(19)般的多重线性表达式的每一项。另外，我们不需要进行模2的处理，因为插值多项式$\binom{x}{11}\binom{15-x}{4}$恰好代表了$x_4\wedge\bar{x}_3\wedge x_2\wedge x_1$。

16. 是的，甚至+也可以，原因是不同的极小项不可能同时为真。（但我们不能在如(25)那样的析取范式上这么做。参见习题35。）

17. 二元运算$\bar{\wedge}$是不满足结合律的，所以一个如$x\,\bar{\wedge}\,y\,\bar{\wedge}\,z$那样的表达式必须被诠释成一个三元（ternary）运算。如果你知道NAND是一个n元运算，那么Quick的记法也是可以的。要注意，单（*single*）变量x的NAND是$\bar{x}$。

18. 如果不是，可以令$u_1\leftarrow\cdots\leftarrow u_s\leftarrow 1$且令$v_1\leftarrow\cdots\leftarrow v_t\leftarrow 0$，使$f$同时为真和为假。（而且如果我们考虑在一个DNF上重复应用分配律(2)直到它变成一个CNF，我们发现反过来同样是正确的：析取式$v_1\vee\cdots\vee v_t$是由f得来的当且仅当它有一个文字与f的每一个蕴涵项相同，当且仅当它有一个文字与f的每一个基本蕴涵项相同，当且仅当它有一个文字与f的某个DNF的每一个蕴涵项相同。）

19. 包含在0010，0011，0101，0110，1000，1001，1010和1011中的极大子立方体是0*10，0101，*01*，和10**；所以答案是$(w\vee\bar{y}\vee z)\wedge(w\vee\bar{x}\vee y\vee\bar{z})\wedge(x\vee\bar{y})\wedge(\bar{w}\vee x)$（而且这个CNF是最短的。）

20. 正确。相应的极大子立方体被包含在某极大子立方体f'和g'中，而且它们的交集不可能再大了。（这个观察归功于Samson和Mills，他们的论文在下面的习题31中将会被引用。）

21. 由布尔定律（20），一个n元函数f是单调的当且仅当它的$(n-1)$元投影g和h是单调的，且有$g\leqslant h$。因此

$$f=(g\wedge\bar{x}_n)\vee(h\wedge x_n)=(g\wedge\bar{x}_n)\vee(g\wedge x_n)\vee(h\wedge x_n)=g\vee(h\wedge x_n)$$

所以不用取补也可以做到。除非函数恒为常数，常数0和1方可消失。反之，任何由$\wedge$和$\vee$建立起的表达式显然是单调的。

相关术语的说明：严格地说，如果遵循经典的数学语言，我们应当说“单调非减”而非简单的“单调”，一个实变量递减函数同样可以说成是单调的（可参见3.3.2G节中的“运行测验”。）但是“非减”相当地拗口；所以布尔函数研究者们几乎一致同意在关于布尔函数的文章中提及“单调”就自动地含有非减的意思。类似地，

数学术语“正函数”通常指一个函数值超过0的函数；但是作者笔下的“正布尔函数”指的是我们称为单调函数的东西。因为一个单调函数是保序的，因此一些作者也采用术语“保序”（isotone）；但那个词已被物理学家、化学家和音乐学家同时采用。

一个形如 $\bar{x}\vee y$ 的布尔函数，如果变量的某个子集被取补它将变成单调函数，则称其为非蚀（unate）函数。定理Q显然也适用于非蚀函数。

22. g和$g\oplus h$两者都是单调的，而且$g(x)\wedge h(x)=0$。

23. $x\wedge(v\vee y)\wedge(v\vee z)\wedge(w\vee z)$。（推论Q同样适用于单调函数的合取基本式。因此，要解决任何这种类型的问题，只需要应用分配律（2）直到一个$\vee$中没有$\wedge$的出现，然后移除任何包含其他所有变量的子句。）

24. 通过对k归纳，根标记为$\vee$的树给出一个包含 $2^{2^{\lceil k/2\rceil-1}}$ 个长度为 $2^{\lceil k/2\rceil}$ 的基本蕴涵项的函数，而根标记为$\wedge$的树给出包含 $4^{2^{\lfloor k/2\rfloor-1}}$ 个长度为 $2^{\lceil k/2\rceil}$ 的基本蕴涵项的函数。比如，当$k=6$时，在根标记为$\wedge$的树中$4^7=2^{14}$个基本蕴涵项有如下形式：

$$x_{(0t_0 0t_{00} 0t_{000})_2}\wedge x_{(0t_0 0t_{00} 1t_{001})_2}\wedge x_{(0t_0 1t_{01} 0t_{010})_2}\wedge x_{(0t_0 1t_{01} 1t_{011})_2}$$
$$\wedge x_{(1t_1 0t_{10} 0t_{100})_2}\wedge x_{(1t_1 0t_{10} 1t_{101})_2}\wedge x_{(1t_1 1t_{11} 0t_{110})_2}\wedge x_{(1t_1 1t_{11} 1t_{111})_2}$$

其中t不是0就是1。[如果想了解关于布尔函数的更多信息，可以参看D. E. Knuth和R. W. Moore，《Artificial Intelligence》6 (1975), 293—326; V. Gurvich和L. Khachiyan，《Discrete Mathematics》169 (1997), 245—248。]

25. 令a_n为答案。则$a_2=a_3=2$, $a_4=3$且$a_n=a_{n-2}+a_{n-3}$，对于$n>4$。因为在k个变量的情形下，对于基本蕴涵项p_k，当$n>4$时，或者是$p_{n-2}\wedge x_{n-1}$或者是$p_{n-3}\wedge x_{n-2}\wedge x_n$。（这些基本蕴涵项对应了路径图$p_n$的极小顶点覆盖。依习题35的含义，当按字母顺序列出时它们是可去壳的（shellable）的。当p_n是7.1.4节习题15中的Perrin数时有$a_n=(7P_n+10P_{n+1}+P_{n+2})/23$。）

26. (a)令$x_j=[j\in J]$，则有$f(x)=0$且$g(x)=1$。（这个事实就是习题18。）

(b)比如，假定$k\in J\in\mathcal{G}$且 $k\notin\bigcup_{I\in\mathcal{F}}I$，又假定已经通过测试（a）。令$x_j=[j\in J$且$j\neq k]$，则$f(x)=1$且$g(x)=0$，因为每个$J'\in\mathcal{G}$且$J'\neq J$都包含有一个元素$\notin J$。

(c)再次，假定条件(a)已排除。如果，比如说$|J|>|\mathcal{F}|$，令$x_j=[\,j$是$I\cap J$中的最小元素，对于某$I\in\mathcal{F}]$。那么有$f(x)=1$，$g(x)=0$。

(d)现在假定 $\bigcup_{I\in\mathcal{F}}I=\bigcup_{J\in\mathcal{G}}J$，每个$I\in\mathcal{F}$代表$f(x)=0$的$2^{n-|I|}$个向量；类似地，每个$J\in\mathcal{G}$代表$g(x)=1$的$2^{n-|J|}$个向量。如果其和$s$少于$2^n$，我们可以计算$s=s_0+s_1$，其中$s_0$计算当$x_n=0$时对$s$的贡献。如果$s_0<2^{n-1}$，令$x_n\leftarrow 0$；否则$s_1<2^{n-1}$，因此我们令$x_n\leftarrow 1$。然后令$n\leftarrow n-1$，最后所有$x_j$都成为已知，并且$f(x)=1$，$g(x)=0$。

27. 令$m=\min(\{|I|\mid I\in\mathcal{F}\}\cup\{|J|\mid J\in\mathcal{G}\})$为最短基本子句或基本蕴涵项的长度。那么$N\cdot 2^{n-m}\geqslant\sum_{I\in\mathcal{F}}2^{n-|I|}+\sum_{J\in\mathcal{G}}2^{n-|J|}\geqslant 2^n$；所以我们有$m\leqslant\lg N$。如果，比如说$|I|=m$，因为每个$J$与$I$相交，某指标$k$出现在$J\in\mathcal{G}$成员中的概率至少为$1/m$。这个观察也证明了提示。

现在令$A(0)=A(1)=1$且对于$v>1$，$A(v)=1+A(v-1)+A(\lfloor\rho v\rfloor)$。那么$A(|\mathcal{F}||\mathcal{G}|)$是递归调用次数的一个上界（X1被执行的次数）。令$B(v)=A(v)+1$，对于$v>1$我们有$B(v)=B(v-1)+B(\lfloor\rho v\rfloor)$，因此对于$v>k$有$B(v)\leqslant B(v-k)+kB(\lfloor\rho v\rfloor)$。取$k=v-\lfloor\rho v\rfloor$以

使$B(v)\leqslant((1-\rho)v+2)B(\lfloor\rho v\rfloor)$，则当$\rho^t v\leqslant 1$也就是当$t\geqslant \ln v/\ln(1/\rho)=\Theta((\log v)(\log N))$时有$B(v)=O(((1-\rho)v+2)^t)$。因此$A(|\mathcal{F}||\mathcal{G}|)\leqslant A(N^2/4)=N^{O(\log N)^2}$。

在实际中，算法运行的速度比上面推导的悲观的上界要快得多。因为一个函数的基本子句是其对偶函数的基本蕴涵项，这个问题实际上和证明一个给定的DNF是另一个的对偶是一样的。而且，如果以$f(x)=0$开始，反复寻找使$f(x)=g(\bar{x})=0$的极小x，我们可以"增长"f直到得到了g的对偶为止。

这里所呈现的思想要归功于M. L. Fredman和L. Khachiyan，《J. Algorithms》21（1996），618—628，他们也提出了算法的改进，并将运行时间减少到了$N^{O(\log N/\log\log N)}$。是否存在多项式时间的算法目前还不知道；但问题不会是NP完全问题，我们可以在低于指数时间内解决它。

28. 结果显然是可以理解的，但是记法和术语令人迷惑；所以我们考虑一个具体的例子：比如说，如果$y_1=y_4=y_6=1$且其他的y_k为0，则函数g是真的当且仅当基本蕴涵项P_1、P_4和P_6覆盖了所有f为真的区域。这样我们看到g的每个蕴涵项和f的每个仅包含基本蕴涵项P_j的DNF之间有一个一一对应的关系。在这种对应下，g的基本蕴涵项对应于"非冗余"DNF，其中没有一个P_j可被省去。

R. B. Cutler和S. Muroga在《IEEE Transactions》C-36 (1987), 277—292中讨论了这个原理的多种精炼技术。

29. B1. [初始化。] 令$k\leftarrow k'\leftarrow 0$。（类似的方法曾在习题5-19中讨论过）

B2. [找出一个零点。] k增加0次或更多次，直到或者$k=m$（停止）或者$v_k\&2^j=0$。

B3. [使$k'>k$。] 如果$k'\leqslant k$，令$k'\leftarrow k+1$。

B4. [推进k'。] k'增加0次或更多次，直到或者$k'=m$（终止）或者$v_{k'}\geqslant v_k+2^j$。

B5. [跳过一个大的不匹配。] 如果$v_k\oplus v_{k'}\geqslant 2^{j+1}$，令$k\leftarrow k'$并且返回B2。

B6. [记录匹配。] 如果$v_{k'}=v_k+2^j$，输出(k,k')。

B7. [推进k。]令$k\leftarrow k+1$，并返回B2。▮

（步骤B3和B5是可选的，但是推荐执行。）

30. 以下算法使用一个可变长度的有序列表，存放于大小不超过$2m+n$的栈S之中。当栈的顶端入口是$S_t=s$时，顶端的列表就是有序集$S_s<S_{s+1}<\cdots<S_{t-1}$。标签位被保持在另一个栈$T$中，它与$S$有同样的大小（在初始化以后）。

P1. [初始化。] 对于$0\leqslant k<m$令$T_k\leftarrow 0$。然后对于$0\leqslant j<n$，应用习题29中的j-伙伴扫描算法，并且对所有找到的(k,k')对，令$T_k\leftarrow T_k+2^j$，$T_{k'}\leftarrow T_{k'}+2^j$。然后令$s\leftarrow t\leftarrow 0$并重复以下的运算直到$s=m$：如果$T_s=0$，输出子立方体$(0,v_s)$并令$s\leftarrow s+1$；否则令$S_t\leftarrow v_s$，$T_t\leftarrow T_s$，$t\leftarrow t+1$，$s\leftarrow s+1$。最后令$A\leftarrow 0$且$S_t\leftarrow 0$。

P2. [推进A。] （此时栈S包含$\nu(A)+1$个子立方体列表。也就是，如果$A=2^{e_1}+\cdots+2^{e_r}$且$e_1>\cdots>e_r\geqslant 0$，则栈中包含所有的子立方体$(a,b)\subseteq V$的$b$值，这些子立方体的$a$值分别是0，$2^{e_1}$，$2^{e_1}+2^{e_2}$，…，$A$，而那些标签为0的子立方体不会出现。所有这些列表都是非空的，除了最后一个有可能不是。我们现在将增加A到下个相关值。）令$j\leftarrow 0$。如果$S_t=t$（也就是如果最上面的列表为空），增加j 0次或者更多次直到$j\geqslant n$或者$A\&2^j\neq 0$。那么当$j<n$并且$A\&2^j\neq 0$时，令$t\leftarrow S_t-1$，$A\leftarrow A-2^j$，并且令$j\leftarrow$

$j+1$。如果$j\geqslant n$则终止算法，否则令$A\leftarrow A+2^j$。

P3. [产生列表A。] 令$r\leftarrow t, s\leftarrow S_t$；并且应用习题29中的$j$-伙伴扫描算法到$r-s$个数$S_s<\cdots<S_{r-1}$上。对所有找到的$(k, k')$对，令$x\leftarrow(T_k\&T_{k'})-2^j$；并且如果$x=0$，输出子立方体$(A, S_k)$，否则令$t\leftarrow t+1, S_t\leftarrow S_k$，$T_t\leftarrow x$。最后令$t\leftarrow t+1$，$S_t\leftarrow r+1$，并且回到步骤P2中。▮

这个算法部分基于Eugenio Morreale的思想[《IEEE Trans》EC-16(1967), 611—620; 《Proc. ACM Nat. Conf》23(1968), 355—365]。运行时间本质上与mn（对于步骤P1）加上包含于V中的总的子立方体的数目成比例。如果$m\leqslant 2^n(1-\varepsilon)$，并且如果$V$是随机选取的且大小为$m$，习题34表明平均的子立方体总数目最多为平均的极大子立方体数目的$O(\log\log n/\log\log\log n)$倍；因此平均运行时间在大多数情况下与输出结果的平均数目几乎成比例。另一方面，习题32和习题116表明输出的数量可能会很大。

31. (a)令$c=c_{n-1}\cdots c_0$, $c'=c'_{n-1}\cdots c'_0$, $c''=c''_{n-1}\cdots c''_0$。一定有某个$j$满足$c_j\neq *$和$c_j\neq c''_j$；否则$c''\subseteq c$。类似地，一定有某个$k$满足$c'_k\neq *$和$c'_k\neq c''_k$。如果$j\neq k$，则有一个点$x_{n-1}\cdots x_0\in c''$它既不在$c$里也不在$c'$里，因为我们可以令$x_j=\bar{c}_j$和$x_k=\bar{c}'_k$。因此$j=k$，而且$j$的值是唯一确定的。而且容易看到$c'_j=\bar{c}_j$。而且如果$i\neq j$，我们有或者$c_i=*$或者$c_i=c''_i$，并且或者$c'_i=*$或者$c'_i=c''_i$。

(b)这个说法显然是(a)的一个推论。

(c)首先我们证明，只要是在步骤E2中，则E2括号中的备注就为真。当$j=0$时显然是对的。除此以外，令$c\subseteq V$是一个j维立方体，并且假定$c=c_0\cup c_1$，其中c_0和c_1是$j-1$维立方体。在步骤E2的前面执行中，对一些c'_0和c'_1，我们有$c_0\subseteq c'_0\in C$和$c_1\subseteq c'_1\in C$；因此$c\subseteq c'_0\sqcup c'_1$或者$c\subseteq c'_0$或者$c\subseteq c'_1$。对于每种情况，$c$都包含在$C$的某个元素里。

其次，我们证明步骤E3输出的结果恰恰就是包含在V中的极大的j维立方体：令$c\subseteq V$为任何一个k维立方体。如果c是极大的，那么当我们运行步骤E3且$j=k$时c将会在C里，而且它会被输出。如果c不是极大的，它有一个伙伴$c'\subseteq V$，它是当我们运行到步骤E3时包含于某个子立方体$c''\in C$里的一个k维立方体。因为$c\subsetneq c''$，所以一致集$c\sqcup c''$将会是C'的一个$j+1$维立方体，而且c不会被输出。

参考：一致集的概念最初是芝加哥大学的Archie Blake在其博士论文中定义的(1937)；参见《J. Symbolic Logic》3(1938), 93, 112—113。后由Edward W. Samson和Burton E. Mills [《Air Force Cambridge Research Center Tech. Report 54-21》(Cambridge, Mass.: April 1954), 54 pp.]以及W. V. Quine[《AMM》62(1955), 627—631]重新独立发现。由于J. A. Robinson使用其一般形式（但是是关于子句而不是蕴涵项的）作为证明定理的"消解原理"的基础[《JACM》12(1965), 23—41]，这个运算有时也被称为归结式（resolvent）。算法E要归功于Ann C. Ewing、J. Paul Roth和Eric G. Wagner，《AIEE Transactions》，Part 1，80(1961)，450—458。

32. (a)将习题31中$\sqcup$的定义改为对于所有的$a\in A$和$x\in\{0,1\}$，关于4个符号$A=\{0,1,*,\bullet\}$是可结合的、可交换的运算：

$$*\sqcup a=a\sqcup *=a,\quad \bullet\sqcup a=a\sqcup\bullet=x\sqcup\bar{x}=\bullet \quad 和 \quad x\sqcup x=x$$

同时令$h(0)=0$，$h(1)=1$，$h(*)=*$和$h(\bullet)=*$。则按分量计算所得的值$c=h(c_1\sqcup\cdots\sqcup c_m)$是广义的一致集，当且仅当这个子立方体包含在$c_1\sqcup\cdots\sqcup c_m$中 [参见P. Tison，《IEEE Transactions》EC-16(1967)，446—456]。

(b)例如，令$c_j=*^{j-1}1*^{m-j}1*^{j-1}0*^{m-j}$。[最后那部分是多余的。所有的解法都已经

由R. H. Sloan、B.Szörényi和G. Turán在《Electronic Colloquium on Computational Complexity》(2005), Report 23中刻画。]

(c)由(a)，每个基本蕴涵项唯一对应于其所“满足”的蕴涵项的子集。[A. K. Chandra和G. Markowsky，《Discrete Math》24(1978), 7—11]

(d)例如，(b)中的 $(y_1\wedge\bar{x}_1)\vee(y_2\wedge x_1\wedge\bar{x}_2)\vee\cdots\vee(y_m\wedge x_1\wedge\cdots\wedge x_{m-1}\wedge\bar{x}_m)$。[J.-M. Laborde，《Discrete Math》32(1980), 209—212。]

33. (a) $\binom{2^n-2^{n-k}}{m-2^{n-k}}\Big/\binom{2^n}{m}$。(b)我们必须排除 $x_1\wedge\cdots x_{j-1}\wedge\bar{x}_j\wedge x_{j+1}\wedge\cdots x_k$ 也是一个蕴涵项的情况。由包含-排斥原理，答案是

$$\sum_l\binom{k}{l}(-1)^l\binom{2^n-(l+1)2^{n-k}}{m-(l+1)2^{n-k}}\Big/\binom{2^n}{m}$$

当$k=n$时，可简化为 $\binom{2^n-n-1}{m-1}\Big/\binom{2^n}{m}$。

34. (a)我们有$c(m,n)=\sum c_j(m,n)$，其中$c_j(m,n)=2^{n-j}\binom{n}{j}\binom{2^n-2^j}{m-2^j}\Big/\binom{2^n}{m}$是具有$n-j$个文字的蕴涵项的平均数量（使用习题30中的术语，即j维子立方体的平均数量）。显然$c_0(m,n)=m$ 并且

$$c_1(m,n)=\frac{nm(m-1)}{2(2^n-1)}\leqslant\frac{mn}{2}\left(\frac{m}{2^n}\right)\leqslant\frac{1}{2}m$$

类似地，$c_j(m,n)\leqslant m/(2^j j!\,n^{2^j-1-j})$。而且 $p(m,n)=\sum_j p_j(m,n)$，其中我们有

$$p_0(m,n)=2^n\binom{2^n-n-1}{m-1}\Big/\binom{2^n}{m}=m\frac{(2^n-n-1)^{\underline{m-1}}}{(2^n-1)^{\underline{m-1}}}\geqslant m\frac{(2^n-n-m)^{m-1}}{(2^n-m)^{m-1}}$$

$$\geqslant m\left(1-\frac{n}{2^n-m}\right)^m\geqslant m\left(1-\frac{n}{2^n-2^n/n}\right)^{2^n/n}=m\exp\left(\frac{2^n}{n}\ln\left(1-\frac{n^2}{2^n(n-1)}\right)\right)$$

(b)注意到$t=\lfloor\lg\lg n-\lg\lg(2^n/m)+\lg(4/3)\rfloor\leqslant\lg\lg n+O(1)$值很小。我们将重复用到事实 $\binom{2^n-j\cdot 2^t}{m-j\cdot 2^t}\Big/\binom{2^n}{m}<\alpha_{mn}^j$，而且当$j$不是太大时，

$$\binom{2^n-j\cdot 2^t}{m-j\cdot 2^t}\Big/\binom{2^n}{m}=\alpha_{mn}^j(1+O(j^2 2^{2t}/m))$$

确实是一个非常好的近似值。为了证实提示的内容，注意到 $\sum_{j<t}c_j(m,n)/c_t(m,n)=O(tc_{t-1}(m,n)/c_t(m,n))=O(t^2/(n\sqrt{\alpha_{mn}}))=O((\log\log n)^2/n^{1/3})$ 并且 $c_{t+j}(m,n)/c_t(m,n)=O((n/(2t))^j a_{mn}^{2^j-1})$。从而我们有 $c(m,n)/c_t(m,n)\approx 1+\frac{1}{2}\binom{n-t}{t+1}\alpha_{mn}$，其中的第二项决

定了α_{mn}何时处于其变化范围的上部。进一步$\sum_l \binom{n-t}{l}(-1)^l\alpha_{mn}^l\left(1+O\left(\frac{l^2 2^{2t}}{m}\right)\right)=(1-\alpha_{mn})^{n-t}+O(n^2\alpha_{mn}(1+\alpha_{mn})^n 2^{2t}/m)$ 因为$(1+\alpha_{mn})^n=O(e^{n^{1/3}})<<m$，误差项是指数级小的。因此$p(m,n)/c_t(m,n)$渐近值为 $e^{-n\alpha_{mn}}+\frac{1}{2}\left(\frac{n-t}{t+1}\right)\alpha_{mn}e^{-n\alpha_{mn}^2}$。

(c)这里$\alpha_{mn}=2^{-2^t}\approx n^{-1}\ln(t/\ln t)$，所以$c(m,n)/c_t(m,n)=1+O(t^{-1}\log t)$，$p(m,n)/c_t(m,n)=t^{-1}\ln t+\frac{1}{2}t^{-1}\ln t+O(t^{-1}\log\log t)$。在这种情况下，可得结论 $\frac{c(m,n)}{p(m,n)}=\frac{2}{3}\frac{\lg\lg n}{\lg\lg\lg n}\left(1+O\left(\frac{\log\log\log\log n}{\log\log\log n}\right)\right)$。

(d)如果$n\alpha_{mn}\leqslant\ln t-\ln\ln t$，我们有$p(m,n)/c(m,n)\geqslant p_t(m,n)/c(m,n)\geqslant t^{-1}\ln t+O(t^{-1}\log t)^2$。另一方面，如果$n\alpha_{mn}\geqslant\ln t-\ln\ln t$我们有$p(m,n)/c(m,n)\geqslant p_{t+1}(m,n)/c(m,n)\geqslant \frac{1}{2}t^{-1}\ln t+O(t^{-1}\log\log t)$。

[$c(m,n)$和$p(m,n)$的均值以及$c(m,n)$的方差，最初由F. Mileto和G. Putzolu进行研究，参见《IEEE Trans》EC-13（1964），87—92；《JACM》12(1965), 364—375。关于随机布尔函数的蕴涵项、基本蕴涵项和非冗余DNF的详细渐近值已由Karl Weber给出，每个布尔函数$f(x_1,\cdots,x_n)$独立等于1的概率是$p(n)$，参见《Elektronische Informationsverarbeitung und Kybernetik》19(1983), 365—374, 449—458, 529—534。]

35. (a)通过重新排列坐标我们可以假定第p个子立方体是$0^k1^u*^v$，因此$B_p=0^k1^u0^v$，$S_p=1^k0^{u+v}$。因为对于$1\leqslant j\leqslant k$，$*^{j-1}1*^{k-j}1^u*^v$的所有点都被覆盖，通过对p归纳，可知$*^k1^u*^v$的所有点也被覆盖。

(b) $B_j\&S_k$非零时第j和第k个子立方体在每个坐标位置上都不同。另一方面，如果$B_j\&S_k$为零，由(a)知，子立方体k的点$\bar{S}_k$落在先前的子立方体中，因为我们有$\bar{S}_k\supseteq B_j$。

(c)从列表1100, 1011, 0011（每个S_k的位下都有划线）我们得到正交 DNF$(x_1\wedge x_2)\vee(x_1\wedge\bar{x}_2\wedge x_3\wedge x_4)\vee(\bar{x}_1\wedge x_3\wedge x_4)$。

(d)有8种解法；比如(01100,00110,00011,11010,11000)。

(e) (0011000,011000,000110,110010,110000,010011,000011)是个对称的解答，而且有更多的可能性。比如，比特码{110000,011000,001100,000110,000011,110010,011010}的42种置换是去壳的。

[单调布尔函数的一个去壳这个概念是由Michael O. Ball和J. Scott Provan引进的，在《Operations Research》36(1988), 703—715中，他们讨论了许多有意义的应用。]

36. 如果$j<k$，对于某些字符串α,β,γ我们有$B_j=\alpha1\beta$并且$B_k=\alpha0\gamma$。形成序列$x_0=\alpha1\gamma$, $x_1=x'_0$, $\cdots$, $x_l=x'_{l-1}$，其中$x_l=\alpha00^{|\gamma|}$。我们有$f(x_0)=1$因为$x_0\supseteq B_k$，但是$f(x_l)=0$

因为 $x_l \subseteq B'_j$ 。所以字符串x_i使$f(x_i)$=1并且$f(x_{i+1})=\cdots=f(x_l)=0$，位于$B$中。它先于$B_k$并且证明了 $B_j \,\&\, S_k \supseteq 0^{|\alpha|}10^{|\beta|}$ 。

[这个构造和习题35的一部分要归功于E. Boros、Y. Crama、O. Ekin、P. L. Hammer、T. Ibaraki和A. Kogan，《SIAM J. Discrete Math》13(2000), 212—226。]

37. 去壳序($000011, 0011\underline{0}1, 00110\underline{0}, 11\underline{0}10\underline{1}, 11\underline{0}10\underline{0}, 110\underline{00}1, 110\underline{0}0\underline{0}$)可推广到所有$n$。同样存在着不基于去壳的有意义的解法，如循环对称（110∗∗∗，1110∗∗，∗∗110∗，∗∗1110，0∗∗∗11，10∗∗11，111111）。

对于下界，给每个点x指定权重 $w_x = -\prod_{j=1}^{n}(x_{2j-1}+x_{2j}-3x_{2j-1}x_{2j})$ ，并且注意到在任何子立方体中所有x的w_x之和为0或者±1。（当n=1时对于那9个可能子立方体中的每一个都足以证明这个奇怪的事实。）现在选取一组不相交的子立方体来划分集合$F=\{x|f(x)=1\}$，我们有

$$\sum_{C\text{chosen}} 1 \geqslant \sum_{C\text{chosen}} \sum_{x\in C} w_x = \sum_{x\in F} w_x \sum_{C\text{chosen}} [x\in C] = \sum_{x\in F} w_x$$

共有 $\binom{n}{k}2^{n-k}$ 个向量x使得恰有k对$x_{2j-1}x_{2j}$=1且权重非零。它们的权重为$(-1)^{k-1}$，并且它们除了当k=0时都位于F中。因此 $\sum_{x\in F} w_x = \sum_{k>0}\binom{n}{k}2^{n-k}(-1)^{k-1} = 2^n-(2-1)^n$ 。

[参见M. O. Ball和G. L. Nemhauser，《Mathematics of Operations Research》4(1979), 132—143。]

38. 肯定不是；一个DNF是可满足的当且仅当它至少有一个蕴涵项。关于一个DNF的难题是判定它是否是一个恒真（总是为真）。

39. 按先序（preorder）将变量$y_1, \cdots, y_N$同每个内部结点联系起来，使每个树结点恰好对应了F的一个变量。对每个带有孩子(l, r)并且被一个二元运算$\circ$标记的内部结点y，构造出4个3CNF子句$c_{00} \wedge c_{01} \wedge c_{10} \wedge c_{11}$，其中

$$c_{pq} = (y^{\overline{p\circ q}N} \vee l^{pN} \vee r^{qN})$$

并且N表示互补的意思（$x^{0N}=x$且$x^{1N}=\bar{x}$）。这些子句实际上表明$y=l\circ r$；比如，如果$\circ$是$\wedge$，则4个子句为 $(\bar{y} \vee l \vee r) \wedge (\bar{y} \vee l \vee \bar{r}) \wedge (\bar{y} \vee \bar{l} \vee r) \wedge (y \vee \bar{l} \vee \bar{r})$。最后，再加一个子句$(y_1 \vee y_1 \vee y_1)$以使$F$=1。

每一个涉及多个对象的事实均可由包含3个对象的事实复合而成。
……以A将B以价格D卖给C为例，
这一包含4个对象的事实可由两个事实复合而成：
首先，A和C达成某种交易，可命名为E；
其次，交易E是以价格D销售B。
——Charles S. Peirce，《谜语竞猜》(1887)

40. 按照提示，A表示的是‘$u<v \oplus v<u$’，B表示的是‘$u<v \wedge v<w \Rightarrow u<w$’。所以

$A\wedge B$表示的是有一个顶点的线性序$u_1<u_2<\cdots<u_n$。（有$n!$种方法来满足$A\wedge B$）现在C表示q_{uvw}和$u<v<w$相等；所以D表示当$u \not- w$时，u和w在排序中是不连续的。因此$A\wedge B\wedge C\wedge D$是可满足的，当且仅当存在一个线性序使得其中所有不相邻的顶点是不连续的（也就是，其中所有的连续出现的顶点是相邻的）。

41. 解法0：'$[m\leqslant n]$' 就是这样的一个公式，但这不在本道习题的范畴之内。

解法1：令x_{jk}的意思是鸽子j占据洞穴k。则对于$1\leqslant j\leqslant m$，子句为$x_{j1}\vee\cdots\vee x_{jn}$；对于$1\leqslant i<j\leqslant m$和$1\leqslant k\leqslant n$，子句为（$\bar{x}_{ik}\vee\bar{x}_{jk}$）。[参见S. A. Cook和R. A. Reckhow，《J. Symbolic Logic》44(1979), 36—50; A. Haken，《Theoretical Comp. Sci》 39 (1985), 297—308。]

解法2：假定$n=2^t$并且令鸽子j占据洞穴$(x_{j1}\cdots x_{jt})_2$。对于$1\leqslant i<j\leqslant m$，子句$((x_{i1}\oplus x_{j1})\vee\cdots\vee(x_{it}\oplus x_{jt}))$可以如习题39般通过引进辅助子句$(\bar{y}_{ijk}\vee x_{ik}\vee x_{jk})\wedge(y_{ijk}\vee x_{ik}\vee\bar{x}_{jk})\wedge(y_{ijk}\vee\bar{x}_{ik}\vee x_{jk})\wedge(\bar{y}_{ijk}\vee\bar{x}_{ik}\vee\bar{x}_{jk})$，改成CNF形式$(y_{ij1}\vee\cdots\vee y_{ijt})$。相比于解法1中的$\Theta(m^2n)$，这个$CNF$的总大小是$\Theta(m^2\log n)$。如果$n$不是2的幂，另外$O(m\log n)$个规模为$O(\log n)$的子句将排除不适当的值。

42. $(\bar{x}\vee y)\wedge(\bar{z}\vee x)\wedge(\bar{y}\vee\bar{z})\wedge(z\vee z)$。

43. 可能不是，因为每个3SAT问题都可以转变成这种形式。例如，子句$(x_1\vee x_2\vee\bar{x}_3)$可以被$(x_1\vee\bar{y}\vee\bar{x}_3)\wedge(\bar{y}\vee\bar{x}_2)\wedge(y\vee x_2)$替代，其中$y$是一个新的变量（本质上等同于$\bar{x}_2$）

44. 假定$f(x)=f(y)=1$蕴涵$f(x\&y)=1$，并且假设$c=x_1\vee x_2\vee\bar{x}_3\vee\bar{x}_4$是$f$的一个基本子句。则$c'=\bar{x}_1\vee x_2\vee\bar{x}_3\vee\bar{x}_4$不是一个子句；否则$c\wedge c'=\bar{x}_2\vee\bar{x}_3\vee\bar{x}_4$也是一个子句，与最初的假设矛盾。所以，存在一个向量y，满足$f(y)=1$且$y_1=1$, $y_2=0$, $y_3=y_4=1$。类似地，存在一个向量z，满足$f(z)=1$并且$z_1=0$, $z_2=1$, $z_3=z_4=1$。但是此时$f(y\&z)=1$，并且c不是一个子句。对于一个有不同文字数量的子句c，同理可得，只要其中至少有两个文字不是互补的。

45. (a) 一个Horn函数$f(x_1,\cdots,x_n)$是非确切的，当且仅当它不等于确切Horn函数$g(x_1,\cdots,x_n)=f(x_1,\cdots,x_n)\vee(x_1\wedge\cdots\wedge x_n)$。所以$f\leftrightarrow g$是确切Horn函数与非确切Horn函数间的一一对应。(b)如果f是单调的，它的补$\bar{f}$或者恒为1或者是一个非确切Horn函数。

46. 算法C置88对xy于核中：当x=a、b、c、0或1时，后续的符号y可以是除"("外的任何符号。当x=(、*、/、+、-时，可以得到y=(,a,b,c,0,1；同样，当x=(,+或者-时，y=-。最后，以x=)开头的合理的对为)+,)-,)*,)/,))。

47. 算法C依拓扑序将顶点置入核，原因是算法在执行TRUTH$(x_k)\leftarrow 1$之前，所有k的前件都已被确保了。但算法2.2.3T使用的是队列而不是栈，所以它实际上产生的顺序通常与算法C不一样。

48. 令$\perp$为一个新变量，并且通过与这个新变量进行OR运算，将每个非确切Horn子句变为确切的。（比如，'$\bar{w}\vee\bar{y}$'变成'$\bar{w}\vee\bar{y}\vee\perp$'也就是'$\bar{w}\wedge\bar{y}\Rightarrow\perp$'；确切Horn子句则保持不变。）然后应用算法C。最初的子句是不可满足的，当且仅当$\perp$在新子句的核中。当设置TRUTH($\perp$)$\leftarrow$1时，算法就可终止。

(J. H. Quick给出另一种解法：将算法C应用到习题45(a)答案所构建的函数g，因为f是不可满足的当且仅当每个变量x_j都在g的核中。然而，f的非确切子句，如$\bar{w}\vee\bar{y}$，会变成g的许多不同的子句$(\bar{w}\vee\bar{y}\vee z)\wedge(\bar{w}\vee\bar{y}\vee x)\wedge(\bar{w}\vee\bar{y}\vee v)\wedge(\bar{w}\vee\bar{y}\vee u)\wedge\cdots$，每个不在原始子句里的变量都对应一个。所以Quick的建议乍看起来可能简洁，但会产生$\Omega(n)$倍的子句数量。)

49. 我们有$f\leqslant g$当且仅当$f\wedge\bar{g}$是不可满足的，当且仅当$f\wedge\bar{c}$对每个g的子句c是不可满足的。但是$\bar{c}$是文字的一个与（AND）式，所以可以应用习题48。[参见H. Kleine Büning和T. Lettmann，《Aussagenlogik: Deduktion und Algorithmen》(1994)，§5.6，书里包含进一步的研究成果。如，有一个有效地验证g是否是f的“换名”的方法，也就是判断是否存在常量$(y_1,\cdots,y_n)$使得$f(x_1,\cdots,x_n)=g(x_1\oplus y_1,\cdots,x_n\oplus y_n)$。]

50. 参见Gabriel Istrate，《Random Structures & Algorithms》20(2002), 483—506。

51. 如果顶点v被标记为A，引进子句$\Rightarrow A^+(v)$和$\Rightarrow B^-(v)$；如果被标记为B，引进子句$\Rightarrow A^+(v)$和$\Rightarrow B^+(v)$。否则令v有k条外向弧$v\to u_1,\cdots,v\to u_k$。对于$1\leqslant j\leqslant k$，引进子句$A^-(u_j)\Rightarrow B^+(v)$和$B^-(u_j)\Rightarrow A^+(v)$。此外，如果$v$没有被标记为C，引进子句和$A^+(u_1)\wedge\cdots\wedge A^+(u_k)\Rightarrow B^-(v)$和$B^+(u_1)\wedge\cdots\wedge B^+(u_k)\Rightarrow A^-(v)$。所有的强制策略均是这些子句的结果。2.2.3节习题28和它的解答提供了更多的信息。

注意，原则上说，除了相应的图远大于物理世界这个细节外，算法C可以被用来判定国际象棋是否对白方而言是强迫取胜（forced victory）的。

52. 最佳发挥的话，结果（参见习题51）是

n	(a)	(b)	(c)	(d)
2	0赢	玩家1赢	1赢	玩家1赢
3	0赢	玩家0赢	玩家0赢	玩家0赢
4	玩家0赢	玩家0赢	玩家0赢	玩家0赢
5	玩家1赢	画	画	如果玩家0先玩则平局
6	玩家1赢	玩家1赢	如果玩家0先玩则平局	如果玩家0先玩则平局
7	如果玩家0先玩则平局	玩家1赢	如果玩家0先玩则平局	如果玩家0先玩则平局
8	画	画	画	如果玩家0先玩则平局
9	画	画	画	如果玩家0先玩则平局

(这里“1 loses if first”的意思是如果玩家0先玩则游戏是个平局，否则0可以赢。)注释：(a)中玩家1有一个微小的劣势，因为当$x_1\cdots x_n$是一个回文时，$f(x)=0$。这个小的不同点甚至在$n=7$时都会影响到结果。虽然玩家1在盘的左半边玩0的情况会更好些，结果是他在$n=4$时的第一步必须是*1**；而另一种选择*0**，将会是平局。游戏(b)实质上是一场赛跑，看谁可以淘汰最后一个*。在游戏(c)中，一个$x_1\cdots x_n$的随机选取使得$f(x)=1$的概率为$F_{n+2}/2^n=\Theta((\phi/2)^n)$；在游戏(d)中，这个概率更慢地趋向于0，为$\Theta(1/\log n)$。仍然，当$n$=2、5、8和9时，玩家1在(c)中做得会比(d)中好；其他情况则不会更差。

53. (a)她应该在第一天或者第二天或者第三天更换。(b, f)有多个可能性，比如将第二天改到第三天。(c)这种情况在图6中说明，将Desert或Excalibur改为

Aladdin。(d) 将Caesars或Excalibur改为Aladdin。(e) 将Bellagio或Desert改为Aladdin。当然，对于没有在循环(42)中出现的Williams，不用对冲突负责任。

54. 如果x和$\bar{x}$都在S内，那么$u\in S \Leftrightarrow \bar{u}\in S$，因为从$x$到$\bar{x}$、$\bar{x}$到$x$、$x$到$u$、和$u$到$x$的路径存在蕴涵着从$\bar{u}$到$\bar{x}$、$\bar{x}$到$\bar{u}$的路径存在，因此从$u$到$\bar{u}$、从$\bar{u}$到$u$也存在。

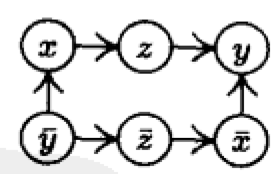
55. (a)成功重命名一个如 $x_1 \vee \bar{x}_2 \vee x3 \vee \bar{x}_4$ 这样的子句的必要且充分条件是 $(y_1 \vee \bar{y}_2) \wedge (y_1 \vee y_3) \wedge (y_1 \vee \bar{y}_4) \wedge (\bar{y}_2 \vee y_3) \wedge (\bar{y}_2 \vee \bar{y}_4) \wedge (y_3 \vee \bar{y}_4)$。变量$\{y_1,\cdots,y_n\}$的长度为2的$\binom{k}{2}$个子句的一个类似集合对应于$\{x_1,\cdots,x_n\}$中的任何长度为$k$的子句。[H. R. Lewis,《JACM》25 (1978), 134—135。]

(b) $\{x_1,\cdots,x_n\}$中一个给定的长度$k>3$的子句通过引进$k-3$个新变量$\{t_2,\cdots,t_{k-2}\}$可以被转换成$3(k-2)$个长度为2的子句，而不是前面的$\binom{k}{2}$个子句，这里举例说明子句$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$的转换：

$$(y_1 \vee y_2) \wedge (y_1 \vee t_2) \wedge (y_2 \vee t_2) \wedge (\bar{t}_2 \vee y_3) \wedge (\bar{t}_2 \vee t_3) \\ \wedge (y_3 \vee t_3) \wedge (\bar{t}_3 \vee y_4) \wedge (\bar{t}_3 \vee y_5) \wedge (y_4 \vee y_5)$$

一般地，对于$1<j<k$，来自$x_1 \vee \cdots \vee x_k$的子句是$(\bar{t}_{j-1} \vee y_j) \wedge (\bar{t}_{j-1} \vee t_j) \wedge (y_j \vee t_j)$，但是用$\bar{y}_1$代替$t_1$并且用$y_k$代替$t_{k-1}$；如果出现了$\bar{x}_j$而非$x_j$，将$y_j$变为$\bar{y}_j$。对给出的每个子句这样处理，对不同的子句使用不同的辅助变量t_j；结果是一个2CNF公式，长度<3m，并且它可满足当且仅当它可重命名为Horn子句。现在应用定理K即得。

[参见B. Aspvall,《J. Algorithms》1(1980), 97—103。由H. Kleine Büning和T. Lettmann记录在《Aussagenlogik：Deduktion und Algorithmen》(1994)中定理5.24的一种结果是任何可满足的2CNF公式可以被重名为Horn子句。注意到同一个函数的两个CNF可能给出不同的结果；比如$(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee z)$，实际上是一个Horn函数，但是这种表现形式的子句不能通过补运算被转换成Horn形式。]



56. 这里$f(x, y, z)$对应了下面展示的有向图（类似于图6），而且它也可以被简化为 $y \wedge (\bar{x} \vee z)$。每个顶点是一个强连通分量。所以公式关于量词∃∃∃,∃∃∀,∀∃∃为真；在∀∃∀, (any)∀(any)这些情况下为假。总之，8种可能性可以在一个立方体的拐角处布置，每次从∃到∀的转变都会使公式为假的可能增大。

57. 形成如定理K中的有向图，我们可以证明量化公式成立当且仅当(i)没有强连通分量同时包含x和$\bar{x}$；(ii)没有路径使一个全称变量x到达另一个全称变量y或者其补$\bar{y}$；(iii) 当‘$\exists v$’出现在‘$\forall x$’的左边时，没有强连通分量同时包含一个全称变量x和存在变量v或者它的补$\bar{v}$。这三个条件显然是必要的，而且当强连通分量被找到时它们是很容易被检验的。

为了说明它们是充分的，首先注意到如果S是一个只存在文字的强连通分量，条件(i)允许我们如定理K般令它们皆是相同的，否则S恰有一个全称文字$u_j=x_j$或者

$u_j=\bar{x}_j$；S中的所有其他文字都是存在文字并且位于x_j的右边，所以我们可以将它们与u_j等同。并且所有的进入S的路径在这种情况下完全地来自于存在强连通分量，它们的值可以被置为0，原因是这种强连通分量的补不能也导向S；因为如果v和$\bar{v}$蕴涵u_j，则u_j蕴涵$\bar{v}$和v。

[参见《Information Proc. Letters》8(1979), 121—123。相比之下，M. Krom已经在J. Symbolic Logic》35(1970), 210—216中证明，一个一阶谓词演算中的类似问题（其中参数化的谓词代替了简单的布尔变量，并且量化是针对参数的）一般来说，实际上是无解的。]

58. 通过如习题48中那样引进'⊥'并且放置'∀⊥'在左边，我们可以假定每个子句是确切的。称全称变量为x_0, x_1, ⋯, x_m（其中x_0是⊥），存在变量为y_1, ⋯, y_n。令'$n \prec v$'表示在量词列表中变量u出现在变量v的左边。当$y_k \prec x_j$时，从任何一个上边没有横线的、文字是y_k的子句中去掉$\bar{x}_j$。这样，对于$0 \leqslant j \leqslant m$，当追加额外子句$(x_0) \wedge \cdots \wedge (x_{j-1}) \wedge (x_{j+1}) \wedge \cdots \wedge (x_m) \wedge \wedge\{(y_k) \mid y_k \prec x_j$并且$y_k \in C_0\}$时，令$C_j$为Horn子句的核（换句话说，$C_j$告诉我们当除了$x_j$外所有的$x$都被假定为真时可以推出什么）。我们断言给出的公式为真当且仅当$x_j \notin C_j$，其中$0 \leqslant j \leqslant m$。

为了证明这个断言，首先注意到如果存在某些j使得$x_j \in C_j$，公式显然为假。（当对于$i \neq j$，$y_k \in C_0$，$y_k \prec x_j$且$x_i=1$时，我们必须令$y^k \leftarrow 1$。）否则我们可以选择每个y_k来使得公式为真，如下：如果$y_k \notin C_0$，令$y^k \leftarrow 0$；否则令$y^k \leftarrow \wedge\{x_j \mid y_k \notin C_j\}$。注意到只有当$x_j \prec x_k$时，$y_k$才依赖于$x_j$。每个包含上边没有横线的文字$x_j$的子句$c$现在为真：因为如果$x_j=0$，对于满足$y_k \notin C_j$的某些$\bar{y}_k$会出现在$c$中，因为$x_j \notin C_j$；因此$y_k=0$。而且每个含有上边没有横线的文字$y_k$的子句$c$也为真：如果$y_k=0$，我们或者有$y_k \notin C_0$，在这种情况下$c$中的某些$y_l \notin C_0$，因此$\bar{y}_l=0$；或者存在某些$j$使得$y_k \notin C_0 \backslash C_j$，在这种情况下某些$x_j=0$，同时或者$\bar{x}_j$出现在$c$中或者某些$\bar{y}_l$出现在$c$中而$y_l \notin C_j$，使得$y_l=0$。

[这种解法要归功于T. Dahlheimer。参见M. Karpinski. H. Kleine Büning和P. H. Schmitt，《Lecture Notes in Comp. Sci》329(1988), 129—137；H. Kleine Büning、K. Subramani和X. Zhao，《Lecture Notes in Comp. Sci》2919(2004), 93—104。]

59. 对n归纳：假定$f(0,x_2,\cdots,x_n)$导向量化结果$y_1,\cdots,y_{2n-1}$，而类似地，$f(1,x_2,\cdots,x_n)$导向量化结果$z_1,\cdots,z_{2^{n-1}}$。那么$\exists x_1 f(x_1,x_2,\cdots,x_n)$导向$y_1 \vee z_1,\cdots,y_{2^{n-1}} \vee z_{2^{n-1}}$，$\forall x_1 f(x_1,x_2,\cdots,x_n)$导向$y_1 \wedge z_1,\cdots,y_{2^{n-1}} \wedge z_{2^{n-1}}$。利用事实$(y \vee z)+(y \wedge z)=y+z$即可得出结论。[参见《Proc. Mini-Workshop on Quantified Boolean Formulas》2 (QBF-02) (Cincinnati:May 2002),1—16。]

60. (a) 和 (b) 皆是。但是 (c) 总是0，(d) 总是1，(e) 是$\overline{\langle xyz \rangle}$，(f) 是$\bar{x} \vee \bar{y} \vee \bar{z}$。

61. 正确，当$w=0$和当$w=1$时确实显然如此。

62. 因为$\{x_1,x_2,x_3\} \subseteq \{0,1\}$，由对称性可以假定$x_1$等于$x_2$。假定$f$只关于头3个变量是单调的，则或者$f(x_1,x_1,x_3,x_4,\cdots,x_n)=f(x_1,x_1,x_1,x_4,\cdots,x_n)$，或者$f(x_1,x_1,x_3,x_4,\cdots,x_n)=f(x_3,x_1,x_3,x_4,\cdots,x_n)$。

63. $\langle xyz \rangle=\langle xxyyzz \rangle$。注释：Emil Post证明了，事实上，任何非平凡单调自对偶

函数的一个单一子例程就足以将它们计算出来。(通过对n归纳，至少有一种适当的方法调用此n元子例程以生成$\langle xyz\rangle$。)

64. [《FOCS》3(1962), 149—157] (a) 如果f是单调且是自对偶的，定理P说明$f(x)=x_k$或者$f(x)=\langle f_1(x)f_2(x)f_3(x)\rangle$。因此条件即刻成立或者可通过归纳证明成立。相反地，如果条件成立，它蕴涵f是单调（当x和y只有一位不同）和自对偶的（当它们所有位均不同）。

(b) 仅需证明在一个新的点上定义f而不引起冲突是可能的。当$f(x)$未定义时，令x为字典序上最小的点。如果$f(\bar{x})$是有定义的，置$f(x)=\overline{f(\bar{x})}$。否则，若对某个$x'\subseteq x$有$f(x')=1$，则置$f(x)=1$；不然置$f(x)=0$。而后条件仍然成立。

65. 如果$\mathcal{F}$是极大相交的，我们有 (i) $X\in\mathcal{F}\Rightarrow\bar{X}\notin\mathcal{F}$，其中$\bar{X}$是$\{1,2,\cdots,n\}/X$的补集；(ii) $X\in\mathcal{F}$且$X\subseteq Y\Rightarrow Y\in\mathcal{F}$，因为$\mathcal{F}\cup\{Y\}$是相交的；还有 (iii) $X\notin\mathcal{F}\Rightarrow\bar{X}\in\mathcal{F}$，因为$\mathcal{F}\cup\{X\}$必须包含一个元素$Y\subseteq\bar{X}$。相反地，可以容易证明任何满足 (i) 和 (ii) 的族$\mathcal{F}$是交叉的，并且如果同时满足 (iii)，则它是极大的。

点睛之笔：若用布尔函数语言，则所有三个阐述都很简单：(i) $f(x)=1\Rightarrow f(\bar{x})=0$; (ii) $x\subseteq y\Rightarrow f(x)\leqslant f(y)$; (iii) $f(x)=0\Rightarrow f(\bar{x})=1$。

66. [T. Ibaraki和T. Kameda,《IEEE Transactions on Parallel and Distributed Systems》4(1993), 779—794]每个具有$Q\subseteq Q'$蕴涵$Q=Q'$这一性质的族明显地对应于一个单调布尔函数f的基本蕴涵项。进一步的条件$Q\cap Q'=\phi$对应于进一步的关系$f(\bar{x})\leqslant\overline{f(x)}$，因为$f(\bar{x})=f(x)=1$成立当且仅当$x$和$\bar{x}$同时使得基本蕴涵项为真。

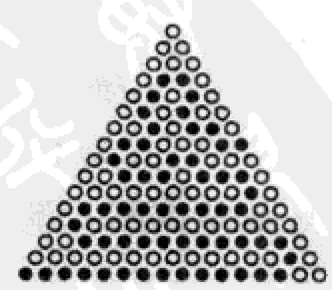
如果仲裁群C和C'以这种方式对应于函数f和f'，则C统治C'当且仅当$f\neq f'$并且对于所有的x有$f'(x)\leqslant f(x)$。那么f'不是自对偶的，因为有一个x使得$f'(\bar{x})=0$，$f(\bar{x})=1$；并且我们有$f(x)=0$，因此$f'(x)=0$。

相反地，如果f'不是自对偶的，则有一个y使得$f'(y)=f'(\bar{y})=0$。如果$y=0\cdots0$，则仲裁群C'是空的，并且被每个其他的仲裁群统治。否则定义$f(x)=f'(x)\vee[x\supseteq y]$。则$f$是单调的，并且对所有的$x$有$f(\bar{x})\leqslant\overline{f(x)}$；所以它对应于一个统治$C'$的仲裁群。

67. (a) 由归纳法，如果$Y(t_{100})=Y(t_{010})=1$，我们就有一个1组成的Y，因为t_{100}和t_{010}内的Y或相交或相邻。类似地，如果$Y(t_{100})=Y(t_{010})=0$我们就没有一个1组成的Y，因为有一个0组成的Y。

(b) 这个公式是由 (a) 和$(t_{abc})_{def}=t_{(a+d)(b+e)(c+f)}=(t_{def})_{abc}$这个事实得出的。[Schensted在1979年1月21日寄给Martin Gardner的一封28页的信中，叙述了这道习题还有习题62和69的结果。Milnor在1957年3月26日给Gardner写了一封信，内容与一种称为“Triangle”的游戏有关。]

68. 当$n=15$时，到目前为止，作者最好的尝试是59个黑石子：($1\leqslant n\leqslant10$的答案分别是2, 3, 4, 6, 8, 11, 14, 18, 23, 27。这些函数的基本蕴涵项可以由相当小的ZDD代表；参见7.1.4节。)



69. 定理P的证明表明了我们仅仅需要证明$Y(T)\leqslant f(x)$即可。在T中的一个Y意味着每个p_j中至少有一个变量。因此$f(\bar{x}_1,\cdots,\bar{x}_n)=0$并且$f(x_1,\cdots,x_n)=1$。

70. 当f是自对偶的，g的自对偶性对于任意的t是显然的。$\overline{g(\bar{x})}=\overline{(f(\bar{x})}\vee[\bar{x}=t]\wedge[\bar{x}\neq\bar{t}]=(f(x)\vee[x=\bar{t}])\wedge[x\neq t]=(f(x)\wedge[x\neq t])\vee(([x=\bar{t}]\wedge[x\neq t]))=g(x)$。

令$x=x_1\cdots x_{j-1}0x_{j+1}\cdots x_n, y=x_1\cdots x_{j-1}1x_{j+1}\cdots x_n$；对于单调性我们必须证明$g(x)\leqslant g(y)$。如果$x=t$或者$y=t$，我们有$g(x)=0$；如果$x=\bar{t}$或者$y=\bar{t}$，我们有$g(y)=1$；其他情况下$g(x)=f(x)\leqslant f(y)=g(y)$[《European J.Combinatorics》16 (1995), 491—501；由J. C. Bioch和T. Ibaraki分别独立发现，《IEEE Transactions on Parallel and Distributed Systems》6(1995), 905—914。]

71. $\langle\langle xyz\rangle uv\rangle=\langle\langle\langle xyz\rangle uv\rangle uv\rangle=\langle\langle\langle yuv\rangle x\langle zuv\rangle\rangle uv\rangle=\langle\langle yuv\rangle\langle xuv\rangle\langle\langle zuv\rangle uv\rangle\rangle=\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle$。

72. 对于（58），$v=\langle uvu\rangle=u$。对于（59），$\langle uyv\rangle=\langle vu\langle xuy\rangle\rangle=\langle\langle vux\rangle uy\rangle=\langle xuy\rangle=y$。对于（60），$\langle xyz\rangle=\langle\langle xuv\rangle yz\rangle=\langle x\langle uyz\rangle\langle vyz\rangle\rangle=\langle xyy\rangle=y$。

73.（a）如果$d(u,v)=d(u,x)+d(x,v)$，我们显然得到一条形式为u—⋯—x—⋯—v的最短路径。相反地，如果$[uxv]$，令u—⋯—x—⋯—v为一条最短路径，其中经过m步到达v，然后经过l步到达x。则$d(u,v)=l+m\geqslant d(u,x)+d(x,v)\geqslant d(u,v)$。

(b)对于所有的z，$\langle zxu\rangle=\langle z\langle vux\rangle\langle yux\rangle\rangle=\langle\langle zvy\rangle ux\rangle\in\{\langle yux\rangle,\langle vux\rangle\}=\{u,x\}$。

(c)我们可以假定$d(x,u)\geqslant d(x,v)>0$。令u—⋯—y—v为一条最短路径，并且令$w=\langle xuy\rangle$。则$\langle vxw\rangle=\langle v\langle vux\rangle\langle wux\rangle\rangle=\langle\langle vvw\rangle ux\rangle=\langle vux\rangle=x$，所以$x\in[w\cdots v]$。我们有$[uwy]$，因为$d(u,y)<d(u,v)$并且$w\in[u\cdots y]$。如果$w\neq u$，我们有$d(w,v)<d(u,v)$；从而$[wxv]$，从而$[uxv]$。如果$w=u$，由（b）我们有$x$—$u$。但是$d(x,u)\geqslant d(x,v)$；因此$x$—$u$，且$[uxv]$。

（d）令$y=\langle uxv\rangle$。由于$y\in[u\cdots x]$，由（a）（c）我们有$d(u,x)=d(u,y)+d(y,x)$。类似地，$d(u,v)=d(u,y)+d(y,v)$，$d(x,v)=d(x,y)+d(y,v)$。但是这三个等式与$d(u,v)=d(u,x)+d(x,v)$一起，得出$d(x,y)=0$。[《Proc. Amer. Math. Soc》12(1961), 407—414。]

74. 由(55)，(55)，(55)，(52)，(51)，(53)和(50)有$w=\langle yxw\rangle=\langle yx\langle zxw\rangle\rangle=\langle yx\langle zx\langle yzw\rangle\rangle\rangle=\langle\langle yxz\rangle x\langle yzw\rangle\rangle=\langle x\langle xyz\rangle\langle wyz\rangle\rangle=\langle\langle xxw\rangle yz\rangle=\langle xyz\rangle$。

75.（a）如果$w=\langle xxy\rangle$，由（iii）有$[xwx]$，从而由（i）有$w=x$。

（b）公理（iii）和（a）告诉我们$[xxy]$总是为真。所以我们可以在（ii）中令$x=y$以推出$[uxv]\Leftrightarrow[vxu]$。因此（iii）中$\langle xyz\rangle$的定义在$x$，$y$和$z$之间是完全对称的。

（c）由（iii）中$\langle uxv\rangle$的定义，我们有$x=\langle uxv\rangle$当且仅当$[uxx]$，$[uxv]$和$[xxv]$。但是我们知道$[uxx]$和$[xxv]$总是为真。

（d）在这一步以及后面的步骤中，我们将建立一个或者更多M的辅助点并且之后使用算法C来得出每个已知的介于关系的结果。（公理有着Horn子句的简单形式。）比如，这里我们定义$z=\langle xyv\rangle$，然后我们可知$[uxy]$，$[uyv]$，$[xzy]$，$[xzv]$和$[yzv]$。从这些假定我们推断出$[uzy]$和$[uzv]$，所以$z=\langle uyv\rangle=y$。

（e）由所提示的构造推出，除了其他之外，还有$[utv]$，$[utz]$，$[vtz]$，$[uwv]$，$[uwz]$，$[vwz]$；因此$t=w$。（在这里使用计算机程序会非常有用。）加上假设$[rws]$，$[rwz]$，$[swz]$，则可得出待证的$[xyz]$；同时可得$r=p$，$s=q$。

（f）令$r=\langle yuv\rangle$，$s=\langle zuv\rangle$，$t=\langle xyz\rangle$，$p=\langle xrs\rangle$，$q=\langle tuv\rangle$；则有$[pqp]$。[《Proc.

Amer. Math. Soc》5(1954),801—807。关于介于关系的公理的早期研究，见E. V. Huntington和J. R. Kline，《Trans.Amer.Math.Soc》18(1917), 301—325。❘

76.公理（i）显然成立，公理（ii）由交换律和（52）得出。基于习题74的答案，我们可从恒等式$\langle xyz\rangle=\langle x\langle xyz\rangle\langle wyz\rangle\rangle$导出（iii）；所以我们仅仅需要验证公式：$\langle x\langle xyz\rangle\langle wyz\rangle\rangle=\langle\langle yxz\rangle x\langle wyz\rangle\rangle=\langle\langle\langle yxz\rangle xz\rangle x\langle wyz\rangle\rangle=\langle\langle yxz\rangle x\langle zx\langle wyz\rangle\rangle\rangle=\langle x\langle xyz\rangle\langle z\langle xyz\rangle w\rangle\rangle=\langle\langle x\langle xyz\rangle z\langle xyz\rangle w\rangle=\langle\langle xyz\rangle\langle xyz\rangle w\rangle$。

注释：最早对中数代数的处理见Birkhoff和Kiss所写的《Bull. Amer.Math.Soc》53(1947), 749—752中，文中假定（50），（51）和短形式的分配律（53）。结合性（52）实际上蕴涵分配性这一事实直到许多年之后才被意识到，M. Kolibiar和T. Marcisová在《Matematický Časopis》24(1974), 179—185中，如本习题一样，通过使用Sholander定理证明了它。R. Veroff和W. McCune使用扩展的Otter定理证明器，于2005年发现了从（50）—（52）到（53）的一个机械推导。

77.（a）在标记的r—s坐标中，假定$l(r)$有个0并且$l(s)$有个1，则在该坐标中，左边的顶点有个0。如果u—v—u'，其中u和u'在左边而v在右边，则$\langle uu'v\rangle$落在左边。但是$[u\cdots v]\cap[u'\cdots v]=\{v\}$，除非$u=u'$。

（b）由推论C，这个说法显然成立。

（c）假定u—v并且u'—v'，其中u和u'在左边，v和v'在右边。令$v=v_0—\cdots—v_k=v'$为一条最短路径，并且令$u_0=u$，$u_k=u'$。由（b），所有的顶点V_j落在右边。左顶点$u_1=\langle u_0u_1u_k\rangle$必须是$u_0$和$v_1$的共同邻居，原因是距离$d(u_0,v_1)=2$。（我们不能得出$u_1=u_0$，因为那样将会推出存在一条从$v$到$v'$并经过左顶点$u$的最短路径。）因此$v_1$的秩为1；类似分析可得，$v_2,\cdots,v_{k-1}$的秩也是如此。[L. Nebeský，《Commentationes Mathematicæ Universitatis Carolinæ》12(1971), 317—325; M. Mulder，《Discrete Math》24(1978), 197—204。]

(d)这些步骤是按照它们到s的距离$d(v,s)$的顺序来访问所有秩为1的顶点的。如果这样的v中有一个还没出现的晚期邻居u，u的秩肯定为1或者2。如果秩为1，u将至少有两个早期邻居，也就是v和将来的MATE(u)。步骤I8将它的判断建立在u的一个随机的早期邻居w的基础上，从而使得$w\neq v$。因为$d(w,v)=2$，由（c），顶点$x=\langle svw\rangle$有秩1。如果w有秩0，那么$x=v$；所以u有秩1。否则$d(x,s)<d(v,s)$，且当x被访问的时候w的秩被正确确定。如果w有秩1，u落在一条从v到w的最短路径上；如果w有秩2，w落在一条从u到s的最短路径上。由（c），在两种情况下u和w有相同的秩。

（e）由（a）和（d），算法移除了所有等同于r—s的边。边的移除显然使图变得不连通了；由（b），剩下的两块是凸的，所以它们是连通的并且实际上它们是中数图。步骤I7记录了两块之间的所有相关关系，因为所有消失的4-循环都在那里被检查。通过对顶点数进行归纳，每一块都被恰当地标记了。

78.每一次当v出现在步骤I4中，它失去了一个邻居u_j。这些v—u_j边中的每一个对应于标记中的一个不同坐标，所以我们可以假定$l(v)$对于某个二进制串α具有形式$\alpha 1^k$。$u_1,u_2,\cdots,u_k$的标记便是$\alpha 01^{k-1},\alpha 101^{k-2},\cdots,\alpha 1^{k-1}0$，通过采取分量中数，我们

现在可以证明所有2^k个形式为$\alpha\beta$的标记会出现在图中的顶点上，因为$\langle\langle\alpha\beta\rangle\langle\alpha\beta'\rangle\langle 0\cdots 0\rangle\rangle$是位串$\alpha(\beta\&\beta')$。

79.（a）如果$l(v)=k$，恰有$v(k)$个更小的顶点是v的邻居。

（b）对于$0\leqslant j<\lceil \lg n\rceil$，最多$\lfloor n/2\rfloor$个1s出现在比特位$j$。

(c)假定恰有k个顶点的标记以0开始。最多$\min(k,n-k)$条边对应于那个比特位，并且最多$f(k)+f(n-k)$条其他的边存在。但是

$$f(n)=\max_{0\leqslant k\leqslant n}(\min(k,n-k)+f(k)+f(n-k))$$

因为函数$g(m,n)=f(m+n)-m-f(m)-f(n)$满足递推关系

$$g(2m+a,2n+b)=ab+g(m+a,n)+g(m,n+b)\quad 0\leqslant a,b\leqslant 1。$$

由归纳可得$g(m,m)=g(m,m+1)=0$，并且当$m\leqslant n$时，$g(m,n)\geqslant 0$。[《Annals of the New York Academy of Sciences》175 (1970), 170—186; D. E. Knuth, 《Proc.IFIP Congress》1971 (1972), 24。]

80.（a）（W. Imrich的解法）顶点标记为0000，0001，0010，0011，0100，0110，0111，1100，1101，1110，1111的图不能用任何本质上不同的方式来标记；但是从0001到1101的距离是4，而不是2。

（b）回路C_{2m}是立方体的一个部分，因为对于$0\leqslant k<m$，它的顶点可以标记为$l(k)=1^k0^{m-k}$，$l(m+k)=0^k1^{m-k}$。但是$l(0)$, $l(m-1)$和$l(m+1)$的按位中数是$01^{m-2}0$；而且当$m>2$时，那些顶点确实没有一个中数。

81. 对。一个中数图是超立方体的一个子图，它是一个二分图。

82. 当G只有两个顶点$\{0,1\}$时，一般情形将简化为简单情形，因为我们可以在中数标记上按分量进行运算，而且$d(u,v)$是$l(u)$和$l(v)$之间的海明距离。

在简单情形中，所述规则使得$u_k\leftarrow v_k$，除了当$u_{k-1}=v_{k-1}=v_{k+1}\neq v_k$时，并且很容易证明它是最佳的。（但是，其他最佳可能性确实存在；比如，如果$v_0v_1v_2v_3=0110$，我们可以令$u_0u_1u_2u_3=0000$。）

[这个问题是出于对自组织数据结构的研究。F. R. K. Chung、R. L. Graham和M. E. Saks在《Discrete Algorithms and Complexity》(Academic Press, 1987), 351—387中，已经证明了中数图是唯一能使u_k总是能被最佳选择为$(v_0, v_1\cdots,v_{k+1})$的一个函数的图，而不用管以后的值$(v_{k+2},\cdots,v_l)$。关于给定一个有限超前量就足够的问题，他们已同样在《Combinatorica》9 (1989), 111—131中刻画了所有情况的特征。]

83. 再一次只要考虑简单的两顶点情形就足够了。在那种情形下的最优解法永远不会有$u_{k-1}\neq u_k$，除了当$u_{k-1}=v_{k-1}\neq v_k=u_k$时。因此我们只需判断$v_0v_1\cdots v_l$中连续0或1的哪个运行会导致$u_0u_1\cdots u_l$中的匹配运行。

假定$v_{k-1}v_k\cdots v_{k+r}=01^r0$并且$u_{k-1}=0$。相比于如果$u_{k-1}u_k\cdots u_{k+r}=01^r0$时2的代价，$u_{k-1}u_k\cdots u_{k+r}=01\cdots 0$的代价是$r\rho$；所以如果$\rho<2/r$我们选择前者，而如果$\rho>2/r$选择后者，这一策略可以用中数的观点来表达，对于$r\geqslant 0$，如果当$2/(r+1)\leqslant\rho<2/r$时，我们令$u_k=\langle u_{k-1}\cdots u_{k-1}v_k\cdots v_{k+r}\rangle$，其中$u_{k-1}$有$r$个事件。（在一个一般中数图中，这个

(2r+1)的中数给出了正确的结果，因为它将会产生有效中数标记；参见习题86。)

但是当接近结束时我们必须改变这条规则，因为它最后要求v_j的值，其中$j>t$。结果是难以处理的；因为如果$v_{k-1}v_k\cdots v_t=01^r$并且$u_{k-1}=0$ ($r=t+1-k$)，则此时只有当$\rho<1/r$时，$u_{k-1}u_k\cdots u_t=00\cdots0$这个选择才是更好的。比如，如果$2/4<\rho<2/3$并且$v_0v_1=01$以及$t=1$，我们将会选择$u_0u_1=00$；但是如果$v_0v_1v_2=011$并且$t=2$，我们将会选择$u_0u_1u_2=011$。一个解法是对于$1\leqslant j\leqslant\lceil r/2\rceil$置$v_{t+j}\leftarrow v_t$并且对于$\lceil r/2\rceil<j\leqslant r$置$v_{t+j}\leftarrow v_0$。

84. 有81个这样的函数，其中的每一个可以被表示为有奇数个元素的中数。7种类型的顶点为：

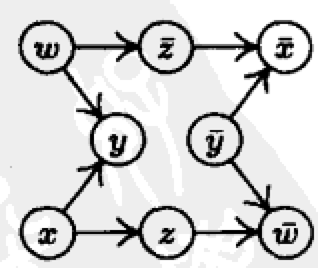
类型	典型顶点	数目	邻接于	度
1	$\langle z\rangle$	5	$\langle vwxyzzz\rangle$	1
2	$\langle vwxyzzz\rangle$	5	$\langle z\rangle,\langle wxyzz\rangle$	5
3	$\langle wxyzz\rangle$	20	$\langle vwxyzzz\rangle,\langle vwxxyyzzz\rangle$	4
4	$\langle vwxxyyzzz\rangle$	30	$\langle xyz\rangle,\langle wxyzz\rangle,\langle vwxyyzz\rangle$	5
5	$\langle vwxyyzz\rangle$	10	$\langle vwxxyyzzz\rangle,\langle vwxyz\rangle$	7
6	$\langle vwxyz\rangle$	1	$\langle vwxyyzz\rangle$	10
7	$\langle xyz\rangle$	10	$\langle vwxxyyzzz\rangle$	3

[Von Neumann和Morgenstern在他们的书籍《Theory of Games and Economic Behavior》(1944)，§52.5中列举了这7种类型，其与一个他们称之为简单游戏(simple games)的获胜联盟系统的等价问题的研究有关。6元函数的图有2646个具有30种类型的顶点，在习题70引用的Meyerowitz的论文中出现过。那些类型中只有21种可以被表示为一个简单的奇数中数；比如，一个如$\langle\langle abd\rangle\langle ace\rangle\langle bcf\rangle\rangle$的顶点就没有这种表示。令$n$元的对应图有$M_n$个顶点；P. Erdös和N. Hindman在《Discrete Math》48 (1984), 61—65中，证明了$\lg M_n$渐近到$\binom{n-1}{\lfloor n/2\rfloor}$。D. Kleitman在《J. Combin. Theory》1 (1966), 153—155中，证明了不同的投影函数的顶点如x和y，在这个图中总是相隔最远的。]

85. 每个强连通分量肯定由一个单点构成；否则两个坐标将总是相等，或总是互补的。这样有向图必是非循环的。

而且，从一个顶点到它的补肯定不存在路径；否则一个坐标将会是常量。

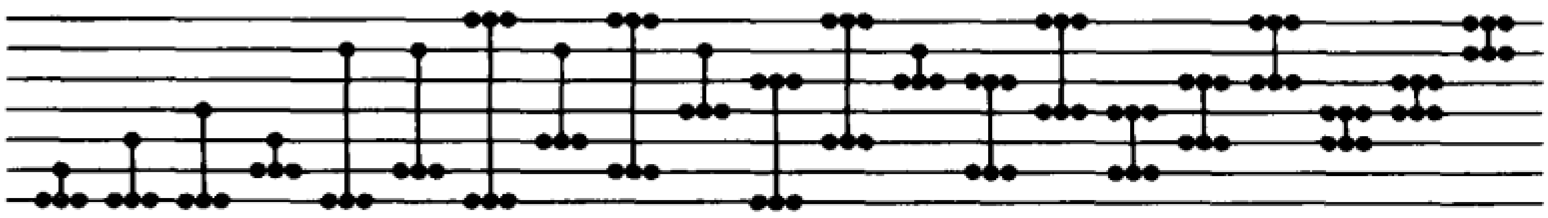
当这两个条件满足时，通过指定值0到所有在x或$\bar{x}$之前的顶点，指定1到所有之后的顶点，并且给其他所有顶点恰当的值，我们可以证明没有顶点x是多余的。



(结果我们得到一种完全不同的方法来表示一个中数图。比如，所画出的有向图对应于标记为{0000，0001，0010，0011，0111，1010}的中数图。)

86.对。由定理P，任何单调自对偶函数将X的元素映入X。

87.这里拓扑序$7654321\bar{1}\bar{2}\bar{3}\bar{4}\bar{5}\bar{6}\bar{7}$可以代替(72)；我们得到

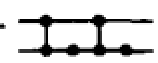


（在同一条线上的连续反转器当然可以被取消。）

88.一个给定的d值至多贡献$6\lceil t/d\rceil$个单位的延迟（对于$2\lceil t/d\rceil$个组）。

89.首先假定新条件是$i\to j$而旧的是$i'\to j'$，其中$i<j$，$i'<j'$，并且没有补文字出现。新模块将$x_1\cdots x_t$变为$y_1\cdots y_t$，其中$y_i=x_i\wedge x_j$，$y_j=x_i\vee x_j$，其他$y_k=x_k$。当$\{i',j'\}\cap\{i,j\}=\varnothing$时我们当然有$y_{i'}\leqslant y_{j'}$。并且如果$i=i'$也没有问题，因为$y_{i'}=y_i\leqslant x_i'=x_{i'}\leqslant x_{j'}=y_{j'}$。但是$i=j'$的情况比较棘手：这里关系$i'\to i$和$i\to j$蕴涵$i'\to j$；并且这个关系已经由先前的模块执行，因为模块已经按照距离d递减的顺序被附加到拓扑顺序$u_1\cdots u_{2t}$中。因此$y_i=x_{i'}\leqslant x_j$并且$y_{i'}\leqslant x_j=x_i$，从而$y_{i'}\leqslant x_{i'}\wedge x_j=y_i=y_{j'}$。当$j=i'$或$j=j'$时，其证明类似。

最后，对于补文字，构造通过反转和不反转比特将一般情况巧妙地简化为非补的情况。

90.当$t=2$时  起作用。一般情况可递归应用该构造块将t减少至$\lceil t/2\rceil$。

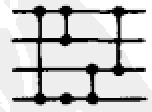
[CI网和其他更普遍的网络的研究是由E. W. Mayr和A. Subramanian开始的，《J. Computer and System Sci》44 (1992), 302—323。]

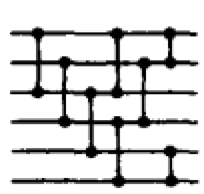
91. 即使在中数图是一棵自由树（有$t+1$个顶点）的特殊情况下，或者是一个如推论F中的分配格的单调情形下，答案也是未知的。在后一种情况中，反转器可能是不必要的。

93. 当u到v的一条最短路径完全落在X中时，令$d_X(u,v)$为该路径的边数。显然$d_X(u,v)\geqslant d_G(u,v)$。并且如果$u=u_0—u_1—\cdots u_k=v$是$G$中的一条最短路径，当$f$是从$G$到$X$的一个回缩映射时，路径$u=f(u_0)—f(u_1)—\cdots f(u_k)=v$在$X$中；因此$d_X(u,v)\leqslant d_G(u,v)$。

94. 如果f是t立方体到X上的一个回缩映射，两个不同的坐标位置不可能总是相等或者对于所有的$x\in X$总是互补的，除非它们是常量。因为如果说X的所有元素具有形式$00*\cdots*$或者$11*\cdots*$，那么两种类型的顶点之间将不存在路径，这与X是一个等距子图（从而连通）的事实矛盾。

给定$x,y,z\in X$；令$w=\langle xyz\rangle$为它们在t立方体中的中数。则$f(w)\in[x\cdots y]\cap[x\cdots z]\cap[y\cdots z]$，因为（举个例子）$f(w)$落在$X$中从$x$到$y$的一条最短路径上。所以$f(w)=w$，并且我们已经证明了$w\in X$ [这个结果和它的相当多的微妙转换要归功于H. J. Bandelt,《J. Graph Theory》8 (1984), 501—510。]

95. 错误（虽然作者希望不是这样）；右边的网络取 $0001\mapsto 0000$，$0010\mapsto 0011$，$1101\mapsto 0110$，但却没有 $\mapsto 0010$。 

（所有可能的输出集合看起来似乎没有一种简单的刻画方式，甚至在没有使用反转器的情况下。比如，  左边的纯比较器网络由Tomás Feder所构建，取 $000000\mapsto 000000$，$010101\mapsto 010101$和$101010\mapsto 011001$，但是没有$\mapsto 010001$。

请参见5.3.4节习题50和5.3.4节习题52。）

96. 不是。如果f是一个建立在实参数$w=(w_1,\cdots,w_n)$和t基础上的阈函数，令$\max\{w\cdot x|f(x)=0\}=t-\varepsilon$。则$\varepsilon>0$，并且$f$是由那$2^n$个当$f(x)=1$时$w\cdot x-t\geqslant 0$，当$f(x)=0$时$t-w\cdot x-\varepsilon\geqslant 0$的不等式所定义的。如果$A$是使得线性不等式系统$Av\geqslant(0,\cdots 0)^T$有一个实值的解$v=(v_1,\cdots,v_N)^T$且$v_N>0$的任何$M\times N$整数矩阵，则同样也会有这样的一个整数解。（通过对$N$归纳证明。）所以我们可以假定$w_1,\cdots,w_n$，$t$和$\varepsilon$是整数。

[一个使用Hadamard不等式的进一步分析（参见4.6.1节等式(25)）证明了整数权重实际上至多$(n+1)^{(n+1)/2}/2^n$就足够了；见S. Muroga、I. Toda和S. Takasu，《J. Franklin Inst.》271(1961)，376—418，定理16。而且，习题112表明了近乎那么大的权重有时候是必须的。]

97. $\langle 11111x_1x_2\rangle$，$\langle 111x_1x_2\rangle$，$\langle 1x_1x_2\rangle$，$\langle 0x_1x_2\rangle$，$\langle 000x_1x_2\rangle$，$\langle 00000x_1x_2\rangle$。

98. 我们可以假设$f(x_1,\cdots,x_n)=\langle y_1^{w_1}\cdots y_n^{w_n}\rangle$，其中权重$w_j$为正整数并且$w_1+\cdots w_n$为奇数。令$\delta$为$2^n$个求和$\pm w_1\pm\cdots\pm w_n$中的最小正值，其中带有$n$个独立的变化符号。记住使得$w_1+\cdots w_k-w_{k+1}-\cdots-w_n=\delta$的所有下标。则 $w_1y_1+\cdots w_ny_n>\frac{1}{2}(w_1+\cdots+w_n)\Leftrightarrow$ $w_1\left(y_1-\frac{1}{2}\right)+\cdots w_n\left(y_n-\frac{1}{2}\right)>0\Leftrightarrow w_1\left(y_1-\frac{1}{2}\right)+\cdots w_n\left(y_n-\frac{1}{2}\right)>-\delta/2\Leftrightarrow w_1y_1+\cdots+w_ny_n>$ $\frac{1}{2}(w_1+\cdots+w_n-(w_1+\cdots+w_k-w_{k+1}-\cdots-w_n))=w_{k+1}+\cdots+w_n\Leftrightarrow w_1y_1+\cdots+w_ky_k-$ $w_{k+1}\bar{y}_{k+1}-\cdots-w_n\bar{y}_n>0$ 。

99. 我们有$[x_1+\cdots x_{2s-1}+s(y_1+\cdots+y_{2t-2})\geqslant st]=[\lfloor(x_1+\cdots x_{2s-1})/s\rfloor+y_1+\cdots+y_{2t-2}\geqslant t]$；并且$\lfloor(x_1+\cdots x_{2s-1})/s\rfloor=[x_1+\cdots x_{2s-1}\geqslant s]$。

（比如，$\langle\langle xyz\rangle uv\rangle=\langle xyzu^2v^2\rangle$，由等式(53)和(54)，我们还知道它是一个等于$\langle x\langle yuv\rangle\langle zuv\rangle\rangle$和$\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle$的量。参考文献：C. C. Elgot，《FOCS》2 (1961)，238。）

100. 正确。由上题及（45）知。

101. (a) 当$n=7$时它们是$x_7\wedge x_6$，$x_6\wedge x_5$，$x_7\wedge x_5\wedge x_4$，$x_6\wedge x_4\wedge x_3$，$x_7\wedge x_5\wedge x_3\wedge x_2$，$x_6\wedge x_4\wedge x_2\wedge x_1$，$x_7\wedge x_5\wedge x_3\wedge x_1$，并且一般来说有$n$个基本蕴涵项，形成一个类似的模式。（我们有或者$x_n=x_{n-1}$，或者$x_n=\bar{x}_{n-1}$。在第一种情况下，$x_n\wedge x_{n-1}$显然是一个基本蕴涵项。在第二种情况下，$F_n(x_1,\cdots,x_{n-1},\bar{x}_{n-1})=F_{n-1}(x_1,\cdots,x_{n-1})$；所以我们用后者的基本蕴涵项，并且当$x_{n-1}$没有出现时插入$x_n$。）

(b) $n=7$的去壳模式是（0000011，000011$\underline{0}$，00011$\underline{0}$1，0011$\underline{0}$1$\underline{0}$，011$\underline{0}$1$\underline{0}$1，11$\underline{0}$1$\underline{0}$1$\underline{0}$，1$\underline{0}$1$\underline{0}$1$\underline{0}$1），其对于所有的n都有效。

(c) 当$n=7$时的几种可能性中的两个说明了一般的情形：

$$F_7(x_1,\cdots,x_7)=Y\begin{pmatrix} & & & & & x_6 & & & & & \\ & & & & x_7 & & x_5 & & & & \\ & & & x_6 & & x_6 & & x_4 & & & \\ & & x_7 & & x_5 & & x_7 & & x_3 & & \\ & x_6 & & x_6 & & x_4 & & x_6 & & x_2 & \\ x_7 & & x_5 & & x_7 & & x_3 & & x_7 & & x_1 \end{pmatrix}=Y\begin{pmatrix} & & & & & x_6 & & & & & \\ & & & & x_7 & & x_5 & & & & \\ & & & x_6 & & x_6 & & x_4 & & & \\ & & x_7 & & x_5 & & x_5 & & x_3 & & \\ & x_6 & & x_6 & & x_4 & & x_4 & & x_2 & \\ x_7 & & x_5 & & x_5 & & x_3 & & x_3 & & x_1 \end{pmatrix}$$

[斐波那契阈函数是由S. Muroga引进的，他同时也发现了习题105中的最优结果；参见《IEEE Transactions》EC-14(1965), 136—148。]

102. (a) 由 (1) 和 (2)，$\hat{f}(\bar{x}_0,\bar{x}_1,\cdots,\bar{x}_n)$是$\hat{f}(x_0,x_1,\cdots,x_n)$的补。

(b) 如果f是由 (75) 给出的，$\hat{f}$是$[(w+1-2t)x_0+w_1x_1+\cdots+w_nx_n\geqslant w+1-t]$，其中$w=w_1+\cdots+w_n$。相反地，如果$\hat{f}$是一个阈函数，那么$f(x_1,\cdots,x_n)=\hat{f}(1,x_1\cdots,x_n)$也是。[E. Goto和H. Takahasi，《Proc. IFIP Congress》(1962), 747—752。]

103. [参见R. C. Minnick，《IRE Transactions》EC-10 (1961), 6—16。]我们想要对于限制$w_j\geqslant 0$ $(1\leqslant j\leqslant n)$ 和对每个基本蕴涵项$x_1^{e_1}\wedge\cdots x_n^{e_n}$有$(2e_1-1)w_1+\cdots+(2e_n-1)w_n\geqslant 1$的条件下求$w_1+\cdots+w_n$的最小值。比如，如果$n=6$，基本蕴涵项$x_2\wedge x_5\wedge x_6$将会导出约束$-w_1+w_2-w_3-w_4+w_5+w_6\geqslant 1$。如果最小值为$+\infty$，则给出的函数不是一个阈函数。(习题84给出了这种情况下的一个最简单的例子。) 此外，如果解$(w_1,\cdots,w_n)$只涉及整数，它使所需的大小最小化。当非整数解出现时，额外的限制必须添加，直到找到最好的解答，正如后面习题的 (c) 部分所述。

104. 首先，当$w_1\geqslant\cdots\geqslant w_n$并且$w_1+\cdots+w_n$是奇数时，我们需要一个算法来生成一个给定的众数函数$\langle x_1^{w_1}\cdots x_n^{w_n}\rangle$的基本蕴涵项$x_1^{e_1}\wedge\cdots\wedge x_n^{e_n}$：

K1. [初始化。] 令$t\leftarrow 0$。然后对于$j=n$，$n-1$，…，1（以此顺序），令$a_j\leftarrow t$，$t\leftarrow t+w_j$，$e_j\leftarrow 0$。最后令$t\leftarrow(t+1)/2$，$s_1\leftarrow 0$和$l\leftarrow 0$。

K2. [输入水平l。] 令$l\leftarrow l+1$，$e_l\leftarrow 1$，$s_{l+1}\leftarrow s_l+w_l$。

K3. [低于阈值？] 如果$s_{l+1}<t$，返回到K2。

K4. [访问一个基本蕴涵项。] 访问指数$(e_1,\cdots,e_n)$。

K5. [缩小规模。] 令$e_l\leftarrow 0$。则如果$s_l+a_l\geqslant t$，令$s_{l+1}\leftarrow s_l$并且回到K2。

K6. [返回。] 令$l\leftarrow l-1$。如果$l=0$则终止；否则如果$e_l=1$则回到K5；否则重复此步骤。▌

(a) $\langle x_1x_2^2x_3^3x_4^5x_5^6x_6^8x_7^{10}x_8^{12}\rangle$ (21个基本蕴涵项)。

(b) 对于$0\leqslant t\leqslant 8$，$\langle x_0^{16-2t}x_1^8x_2^4x_3^2x_4\rangle$的最优权重是$w_0w_1w_2w_3w_4$=10000，31111，21110，32211，11100，23211，12110，13111，01000；其他情况下是对偶的。

(c) 这里最优权重$(w_1,\cdots,w_{10})$为(29,25,19,15,12,8,8,3,3,0)/2；所以我们认识到x_{10}是不相关的，并且我们必须处理分数的权重。限制$w_8\geqslant 2$得到整数权重 (15，13，10，8，6，4，4，2，1，0)，它是最优的是因为它们的总和比先前的总和多2。(175 428个9元自对偶阈函数中只有两个有非整数最小权重$w_1+\cdots+w_n$；另一个是$\langle x_1^{17}x_2^{15}x_3^{11}x_4^9x_5^7x_6^5x_7^4x_8^2x_9\rangle$。一个最小值表达式中的最大的$w_1$出现在$\langle x_1^{42}x_2^{22}x_3^{18}x_4^{15}x_5^{13}$

$x_6^{10}x_7^8x_8^4x_9^3\rangle$ 中；最大的$w_1+\cdots+w_9$唯一地出现在$\langle x_1^{34}x_2^{32}x_3^{28}x_4^{27}x_5^{24}x_6^{20}x_7^{18}x_8^{15}x_9^{11}\rangle$中，它同样也是最大$w_9$的一个例子。参见S. Muroga、T. Tsuboi和C. R. Baugh，《IEEE Transactions》C-19 (1970)，818—825。)

105．当n=7时，习题103中产生的不等式为$w_7+w_6-w_5-w_4-w_3-w_2-w_1\geqslant 1$，$-w_7+w_6+w_5-w_4-w_3-w_2-w_1\geqslant 1$，$w_7-w_6+w_5+w_4-w_3-w_2-w_1\geqslant 1$，$-w_7+w_6-w_5+w_4+w_3-w_2-w_1\geqslant 1$，$w_7-w_6+w_5-w_4+w_3+w_2-w_1\geqslant 1$，$-w_7+w_6-w_5+w_4-w_3+w_2+w_1\geqslant 1$，$w_7-w_6+w_5-w_4-w_3-w_2+w_1\geqslant 1$。

将它们分别乘以1，1，2，3，5，8，5得到$w_1+\cdots+w_7\geqslant 1+1+2+3+5+8+5$。同样的思想对于所有的$n\geqslant 3$都有效。

106.（a）$\langle x_1^{2^{n-1}}x_2^{2^{n-2}}\cdots x_n\bar{y}_1^{2^{n-1}}\bar{y}_2^{2^{n-2}}\cdots\bar{y}_n\bar{z}\rangle$（由习题99，我们也可以执行$n$个3中数的计算：$\langle\langle\cdots\langle x_n\bar{y}_n\bar{z}\rangle\cdots x_2\bar{y}_2\rangle x_1\bar{y}_1\rangle$。)

（b）如果$\langle x_1^{u_1}x_2^{u_2}\cdots x_n^{u_n}\bar{y}_1^{v_1}\bar{y}_2^{v_2}\cdots\bar{y}_n^{v_n}\bar{z}^{w}\rangle$解答了这个问题，$2^{n+1}-1$个基本不等式必须成立；比如，当$n$=2时它们是$u_1+u_2-v_1+v_2-w\geqslant 1$，$u_1+u_2-v_1-v_2+w\geqslant 1$，$u_1-u_2+v_1-v_2-w\geqslant 1$，$u_1-u_2-v_1+v_2+w\geqslant 1$，$-u_1+u_2+v_1+v_2-w\geqslant 1$，$-u_1+u_2+v_1-v_2+w\geqslant 1$，$-u_1-u_2+v_1+v_2+w\geqslant 1$。将它们所有都加起来得到$u_1+u_2+\cdots u_n+v_1+v_2+\cdots+v_n+w\geqslant 2^{n+1}-1$。

107.

f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$
$\bot$	0	(0,0)	$\bar{\subset}$	1	(0,1)	$\bar{\vee}$	1	(0,0)	$\bar{\mathrm{L}}$	2	(0,1)
$\land$	1	(1,1)	R	2	(1,2)	$\equiv$	2	(1,1)	$\supset$	3	(1,2)
$\bar{\supset}$	1	(1,0)	$\oplus$	2	(1,1)	$\bar{\mathrm{R}}$	2	(1,0)	$\bar{\land}$	3	(1,1)
L	2	(2,1)	$\vee$	3	(2,2)	$\subset$	3	(2,1)	$\top$	4	(2,2)

注意到⊕与≡有相同的参数$N(f)$和$\Sigma(f)$；它们是唯一的不是阈函数的布尔二元运算。

108.如果$\Sigma(g)=(s_0,s_1,\cdots,s_n)$，当$x_0$=1时$g$的值在$s_0$种情况下为1，而当$x_0$=0时$g$的值在$2^n-s_0$种情况下为1。我们同样有$\Sigma(f_0)+\Sigma(f_1)=(s_1,\cdots,s_n)$，并且

$$\Sigma(f_0)=\sum_{x_1=0}^{1}\cdots\sum_{x_n=0}^{1}(\bar{x}_1,\cdots,\bar{x}_n)g(0,\bar{x}_1,\cdots,\bar{x}_n)$$

$$=\sum_{x_1=0}^{1}\cdots\sum_{x_n=0}^{1}((1,\cdots,1)-(x_1,\cdots,x_n))(1-g(1,x_1,\cdots,x_n))$$

$$=(2^{n-1}-s_0,\cdots,2^{n-1}-s_0)+\Sigma(f_1)$$

所以对于$n>0$，答案是（a）$N(f_0)=2^n-s_0$，$\Sigma(f_0)=\frac{1}{2}(s_1-s_0+2^{n-1},\cdots,s_n-s_0+2^{n-1})$；（b）$N(f_1)=s_0$，$\Sigma(f_1)=\frac{1}{2}(s_1+s_0+2^{n-1},\cdots,s_n-s_0-2^{n-1})$。

[1963年E. Goto在MIT的演讲中给出了等价的结果。]

109.（a）$a_1+\cdots+a_k\geqslant b_1+\cdots+b_k$当且仅当$k-a_1-\cdots-a_k\leqslant k-b_1-\cdots-b_k$。

（b）令$\alpha^+=(a_1,a_1+a_2,\cdots,a_1+\cdots+a_n)$。则由$\alpha^+$和$\beta^+$的分量极小化得到的向量$(c_1,\cdots,c_n)$是$(\alpha\land\beta)^+$。(显然$c_j=c_{j-1}+a_j$或$b_j$。)

（c）如（b）中那样进行下去，但是取分量极大化；或者取$\overline{\bar{\alpha}\land\bar{\beta}}$。

(d) 正确，因为max和min满足这些分配律。(实际上，我们可以以类似的方式从所有n元组集$a_1\cdots a_n$（对于$1\leqslant j\leqslant n$有$0\leqslant a_j<m_j$）的集合中得到一个可分配的混合-基数的众数格（mixed-radix majorization lattice）。)

(e) α_1包含α_0并且$\alpha 10\beta$包含$\alpha 01\beta$。[这个刻画要归功于R. O. Winder，《IEEE Trans.》EC-14 (1965), 315—325，但是他没有证明格的性质。格经常被称为$M(n)$；参见B. Lindström，《Nordisk Mat. Tidskrift》17 (1969), 61—70；R. P. Stanley，《SIAM J. Algebraic and Discrete Methods》1 (1980), 177—179。]

(f) 因为(e)我们有$r(\alpha)=na_1+(n-1)a_2+\cdots+a_n$。

(g) 关键是$0\beta\succeq 0\alpha$当且仅当$\beta\succeq\alpha$，并且$1\beta\succeq 0\alpha$当且仅当$1\beta\succeq 10\cdots 0\vee 0\alpha=1\alpha'$。

(h) 就是说，有多少$a_1\cdots a_n$具有性质：$a_1\cdots a_k$包含的1不比包含的0多？答案是$\binom{n}{\lfloor n/2\rfloor}$；例如，参见2.2.1节习题4或者7.2.1.6节习题42（a）。

110. (a) 如果$x\subseteq y$则$x\preceq y$，从而$f(x)\leqslant f(y)$；证明完毕。

(b) 不是；一个阈函数不需要是单调的（见(79)）。但是我们可以证明f是正则的如果我们同时要求$w_n\geqslant 0$：因为如果$f(x)=1$并且y包含x则有$w\cdot y\geqslant w\cdot x$。

(c) 只要$f(x)=1$并且$x_j<x_{j+1}$，则当y覆盖$x(x_j\leftrightarrow x_{j+1})$我们有$f(y)=1$；从而$s_j\geqslant s_{j+1}$。(这个论点即使在$w_n<0$时也成立。)

(d) 不是；比如考虑$\langle x_1x_2^2x_3^2\rangle$，它等于$\langle x_1x_2x_3\rangle$。其反例即使在权重$w_1+\cdots+w_n$最小时也会出现，因为习题103中的线性规划的解不总是唯一的。一种同样的情形，由Muroga、Tsuboi和Baugh发现的是$\langle x_1^{17}x_2^9x_3^8x_4^6x_5^7x_6^5x_7^3x_8^2x_9^2\rangle$，一个实际在$x_4$和$x_5$上是对称的函数。但是因为(c)，如果$s_j>s_{j+1}$我们必有$w_j>w_{j+1}$。

111. (a) 如习题14中那样逐点找一个最优自对偶函数f；在平局的情况下，令$f(x_1,\cdots,x_n)=x_1$。则 $f(x_1,\cdots,x_n)=[r_1^{x_1}\cdots r_n^{x_n}\geqslant\sqrt{r_1\cdots r_n}]$，除了当$x_1=0$时，'$\geqslant$'变成'$>$'。当$r_1\geqslant\cdots\geqslant r_n\geqslant 1$时，函数是正则的。

(b) 令g是(a)中构建的正则、自对偶函数。如果f是一个给定的正则、自对偶函数，我们想要证明对于所有的向量x有$f(x)\leqslant g(x)$；这将蕴涵$f=g$，因为两个函数都是自对偶的。

假定$f(x)=1$，并且令$y\preceq x$为满足$f(y)=1$的极小者。如果我们已经证明了$g(y)=1$，则确实如所要求的$g(x)=1$。[参见K. Makino和T. Kameda，《SIAM Journal on Discrete Mathematics》14 (2001), 381—407。]

比如，当$n=5$时只有7个自对偶正则布尔函数，由如下图8中的极小元素产生10000；01111,10001；01110,10010；01101,10011,10100；01100；01011,11000；00111。所以一个最优仲裁群可以通过仅仅检查一些函数值来找出。

(c) 假定$1>p_1\geqslant\cdots p_r\geqslant\frac{1}{2}>p_{r+1}\geqslant\cdots\geqslant p_n>0$。令$f_k(x_1,\cdots,x_n)$为第$k$个单调自对偶函数并且$F_k(x_1,\cdots,x_n)$为它的整系数多重线性表示。我们想要找出最优有效性$G(p_1,\cdots,p_n)=\max_k F_k(p_1,\cdots,p_n)$。如果$p_1\leqslant p'_1,\cdots,p_n\leqslant p'_n$，由习题12(e)我们有$F_k(p_1,\cdots,p_n)\leqslant F_k(p'_1,\cdots,p'_n)$；从而$G(p_1,\cdots,p_n)\leqslant G(p'_1,\cdots,p'_n)$。

因此如果$0<r<n$我们有$G(p_1,\cdots,p_n)\leqslant G\left(p_1,\cdots,p_r,\frac{1}{2},\cdots,\frac{1}{2}\right)$。并且后者是$F\left(p_1,\cdots,p_r,\frac{1}{2},\cdots,\frac{1}{2}\right)$，其中$f$是（a）部分中得到的函数。这个函数不依赖于$(x_{r+1},\cdots,x_n)$，所以它给出了最优解。

如果$r=0$问题看起来似乎更深奥了。我们有$G(p_1,\cdots,p_n)\leqslant G(p_1,\cdots,p_1)$；所以如果我们可以证明对于所有的$k$只要当$p<\frac{1}{2}$时就有$F_k(p,\cdots,p)\leqslant p$，我们就可以推断出最优仲裁群是$f(x_1,\cdots,x_n)=x_1$。一般地，$F_k(p,\cdots p)=\sum_m c_m p^m(1-p)^{n-m}$，其中$c_m$是使得$f_k(x)=1$并且$\nu x=m$的矢量$x$的数目。习题70告诉我们对于所有的$k$，$c_m+c_{n-m}=\binom{n}{m}$。并且Erdös-Ko-Rado定理（7.2.1.3节习题111）告诉我们当$m\leqslant n/2$时，对于所有的m集的相交族有$c_m\leqslant\binom{n-1}{m-1}$。结果随之产生。

[参见Y. Amir和A. Wool，《Information Processing Letters》65 (1998), 223—228。]

112.（a）主要的项分别是0，$+xy$，$-xy$，$+x$，$-xy$，$+y$，$-2xy$，$-xy$，$+xy$，$+2xy$，$-y$，$+xy$，$-x$，$+xy$，$-xy$，1；所以当f是 ∧, ∟, R, $\bar\nabla$, ≡, ⊂, ⊃, ⊤时，$F(f)=1$。

(b) 按照答案12中的记法，对应指数01101的系数，比方说是f_{0**0*}，它是真值表项的一个线性组合，当有k个星号时，总是落在范围$\lceil -2^{k-1}\rceil\leqslant f_{0**0*}\leqslant\lceil 2^{k-1}\rceil$内。这样主系数是正的当且仅当混合基数（mixed-radix）数

$$\left[\begin{matrix} f_{**\cdots*}, f_{0*\cdots*}, \cdots, f_{*0\cdots0}, f_{00\cdots0} \\ 2^m+1, 2^{m-1}+1, \cdots, 2^1+1, 2^0+1 \end{matrix}\right]$$

是正的，其中f是按照与Chase序列相反的次序排列的，并且基数2^k+1对应于一个有k个星号的f。比如当$m=2$时，我们有$F(f)=1$当且仅当总和$18f_{**}+6f_{0*}+2f_{*0}+f_{00}=18(f_{11}-f_{01}-f_{10}+f_{00})+6(f_{01}-f_{00})+2(f_{10}-f_{00})+f_{00}=18f_{11}-12f_{01}-16f_{10}+11f_{00}$是正的；所以阈函数可以写为$\langle f_{11}^{18}\bar{f}_{01}^{12}\bar{f}_{10}^{16}f_{00}^{11}\rangle$。

（在这种特定情况下，更简单的表达式$\langle f_{11}f_{11}\bar{f}_{01}\bar{f}_{10}f_{00}\rangle$实际上是有效的。但是（c）部分将要证明当$m$更大时我们不能做出很大的改善。）

（c）假定$F(f)=\left[\sum_\alpha \nu_\alpha\left(f_\alpha-\frac{1}{2}\right)>0\right]$，其中求和是对所有$2^m$个长度为$m$的二进制串并且其中每个$\nu_\alpha$是一个整数权重。定义$w_\alpha=\sum_\beta(-1)^{\nu(\alpha-\beta)}\nu_\beta$和$F_\alpha=\sum_\beta(-1)^{\nu(\alpha-\beta)}f_\beta-2^{m-1}[\alpha=00\cdots0]$；这样，比如$w_{01}=-\nu_{00}+\nu_{01}-\nu_{10}+\nu_{11}$和$F_{11}=f_{00}-f_{01}-f_{10}+f_{11}$。可以

证明如果$F_\alpha=0$，则$F_{1^k0^l}=2^l f_{*k0^l}$，其中$\nu(\alpha)>k>0$；因此转化后的真理系数的符号F_α决定了多重线性表示中的主系数的符号。而且，我们现在有 $F(f)=\left[\sum_\alpha w_\alpha F_\alpha>0\right]$。

证明的总体思路是要选择测试函数f，从中我们可以得到变换后的权重w_α的性质。比如，如果$f(x_1,\cdots,x_m)=x_1\oplus\cdots\oplus x_k$，我们发现对于所有的$\alpha$除了$F_{1^k0^{m-k}}=(-1)^{k-1}2^{m-1}$外，有$F_\alpha=0$。$x_1\oplus\cdots\oplus x_k$的多重线性表示有主项$\lceil(-2)^{k-1}\rceil x_1\cdots x_k$；从而我们可以推断出$w_{1^k0^{m-k}}>0$，并且由类似的方法得到对于所有的$\alpha$有$w_\alpha>0$。一般地，如果$m$变为$m+1$但是$f$不依赖于$x_{m+1}$，我们有$F_{\alpha0}=2F_\alpha$和$F_{\alpha1}=0$。
测试函数 $x_2\oplus\cdots\oplus x_m\oplus x_1\bar{x}_2\cdots\bar{x}_m$ 证明了

$$w_{1^m}>(2^{m-1}-1)w_{01^{m-1}}+\sum_{k=1}^{m-1}w_{1^k01^{m-1-k}}+\text{较小项}$$

其中的较小项中只涉及w_α和$\nu(\alpha)\leqslant m-2$。特别地，$w_{11}>w_{01}+w_{10}+w_{00}$。测试函数 $x_1\oplus\cdots\oplus x_{m-1}\oplus\bar{x}_1\cdots\bar{x}_{m-2}(x_{m-1}\oplus\bar{x}_m)$证明了

$$w_{1^{m-2}01}>(2^{m-2}-1)w_{1^{m-2}10}+\sum_{k=0}^{m-3}(w_{1^k01^{m-3-k}10}+w_{1^k01^{m-3-k}01})+\text{较小项}$$

其中这次的较小项有$\nu(\alpha)\leqslant m-3$。特别地，$w_{101}>w_{110}+w_{010}+w_{001}$。通过重排下标，我们得到类似的不等式导出

$$w_{\alpha_j}>(2^{\nu(\alpha_j)-1}-1)w_{\alpha_{j-1}}，\text{对于}0<j<2^m$$

因为w开始迅速增长。但是我们有 $\nu_\alpha=\sum_\beta(-1)^{\nu(\beta-\alpha)}w_\beta/n$；从而$|\nu_\alpha|=w_{11\ldots1}/n+O(w_{11\ldots1}/n^2)$。[《SIAM J. Discrete Math》7 (1994), 484—492。这一结果的重要概括已经由N. Alon和V. H. Vũ取得，《J. Combinatorial Theory》A79 (1997), 133—160。]

113. 所述的g_3是$S_{2,3,6,8,9}$，因为所述的g_2是$S_{2,3,4,5,8,9,10,11,12}$。

对于更难的函数$S_{1,3,5,8}$，令$g_1=[\nu x\geqslant 6]$；$g_2=[\nu x\geqslant 3]$；$g_3=[\nu x-5g_1-2g_2\geqslant 2]=S_{2,4,5,9,10,11,12}$；$g_4=[2\nu x-15g_1-9g_3\geqslant 1]=S_{1,3,5,8}$。[参见M. A. Fischler和M. Tannenbaum，《IEEE Transactions》C-17 (1968), 273—279。]

114. $[4x+2y+z\in\{3,6\}]=(\bar{x}\wedge y\wedge z)\vee(x\wedge y\wedge\bar{z})$。按照同样的方法，任何$n$元的布尔函数是一个$2^n-1$元对称函数的一个特殊情形。[参见W. H. Kautz，《IRE Transactions》EC-10 (1961), 378。]

115. 两边都是自对偶的，所以我们可以假定$x_0=0$。则

$$s_j=[x_j+\cdots+x_{j+m-1}>x_{j+m}+\cdots+x_{j+2m-1}]$$

如果$x_1+\cdots+x_{2m}$是奇数，我们有$s_j=\bar{s}_{j+m}$；从而$s_1+\cdots+s_{2m}=m$并且结果为1。但是如果$x_1+\cdots+x_{2m}$是偶数，则至少存在一个$j\leqslant m$，使得差异$x_j+\cdots+x_{j+m-1}-x_{j+m}-\cdots-x_{j+2m-1}$将会是0；这使得$s_j=s_{j+m}=0$，所以我们将有$s_1+\cdots+s_{2m}<m$。

116. (a) 它是一个蕴涵项当且仅当只要$j\leqslant\nu x\leqslant n-k+j$就有$f(x)=1$。它是一个基

本蕴涵项当且仅当只要$\nu x=j-1$或者$\nu x=n-k+j+1$就有$f(x)=0$。

(b) 考虑串$v=v_0v_1\cdots v_n$使得$f(x)=v_{\nu x}$。由(a)部分，当$v=0^a1^{b+1}0^c$时有$\binom{a+b+c}{a,b,c}$个基本蕴涵项。在所述情形下，$a=b=c=3$，所以有1680个基本蕴涵项。

(c) 对于一个一般的对称函数，我们将v中1的每次运行的基本蕴涵项加在一起。显然当$a<c-1$时，$v=0^{a+1}1^{b+1}0^{c-1}$的情况比$v=0^a1^{b+1}0^c$更多；所以当达到最大值时，v不包含两个连续的0。

对于$m<j\leqslant n$，当$v_m=1$且$v_j=0$时，令$\hat{b}(m,n)$为基本蕴涵项可能数量的最大值。则当$m\leqslant \frac{1}{2}n$时，我们有

$$\hat{b}(m,n)=\max_{0\leqslant k\leqslant m}\left(\binom{n}{k,m-k,n-m}+\hat{b}(k-2,n)\right)$$
$$=\binom{n}{\lceil m/2\rceil,\lfloor m/2\rfloor,n-m}+\hat{b}(\lceil m/2\rceil-2,n)$$

和$\hat{b}(-2,n)=\hat{b}(-1,n)=0$。并且全部的最大值为

$$\hat{b}(n)=\binom{n}{n_0,n_1,n_2}+\hat{b}(n_1-2,n)+\hat{b}(n_2-2,n),n_j=\left\lfloor\frac{n+j}{3}\right\rfloor$$

特别地，我们有$\hat{b}(9)=1698$，并且最大值发生在$v=1101111011$的时候。

(d) 由Stirling近似式，我们有$\hat{b}(n)=3^{n+3/2}/(2\pi n)+O(3^n/n^2)$。

(e) 在这个情形中对于$m<\lceil n/2\rceil$的适当的递推式是

$$\tilde{b}(m,n)=\max_{0\leqslant k\leqslant m}\left(\binom{n}{k,m-k,n-m}+\binom{n}{k-1,0,n-k+1}+\tilde{b}(k-2,n)\right)$$
$$=\binom{n}{\lceil m/2\rceil,\lfloor m/2\rfloor,n-m}+\binom{n}{\lceil m/2\rceil-1}+\tilde{b}(\lceil m/2\rceil-2,n)$$

和$\tilde{b}(n)=\tilde{b}(\lceil n/2\rceil-1,n)$最大化min（基本蕴涵项$(f)$，基本蕴涵项$(\bar{f})$）。

我们有$(\tilde{b}(1),\tilde{b}(2),\cdots)=(1,1,4,5,21,31,113,177,766,1271,4687,7999,34412,\cdots)$；比如，$\tilde{b}(9)=766$对应于$S_{0,2,3,4,8}(x_1,\cdots,x_9)$。渐近地，$\tilde{b}(n)=2^{(3n+3+(n \bmod 2))/2}/(2\pi n)+O(2^{3n/2}/n^2)$。

参考文献：Summaries，《Summer Inst. for Symbolic Logic》(《Dept. of Math.》, Cornell Univ., 1957), 211—212; B. Dunham和R. Fridshal，《J. Symbolic Logic》24 (1959), 17—19; A. P. Vikulin，《Problemy Kibernetiki》29 (1974), 151—166，它报告了1960年所完成的工作；Y. Igarashi，《Transactions of the IEICE of Japan》E62 (1979), 389—394。

117. 当我们选择所有维数为$\lfloor n/3\rfloor$的子立方体时，我们得到n立方体的不互相

包含的子立方体的个数的最大值。（它也可以由选择所有维数为 $\lfloor(n+1)/3\rfloor$ 的子立方体得到；比如当n=2时，我们可以或者选择{0*, 1**0, *1}或者选择{00, 01, 10, 11}。）从而 $b*(n)=\binom{n}{\lfloor n/3\rfloor}2^{n-\lfloor n/3\rfloor}=3^{n+1}/\sqrt{4\pi n}+O(3^n/n^{3/2})$ [见前面答案中Vikulin的论文，164—166页；A. K. Chandra和G. Markowsky，《Discrete Math.》24 (1978), 7—11; N. Metropolis和G. C. Rota，《SIAM J. Applied Math.》35 (1978), 689—694。]

118. 考虑两个函数相等，如果我们可以通过取补或者交换变量从一个得到另一个，但不是对函数值本身取补。这样的函数显然有相同数目的基本蕴涵项；这种相等关系在下面的答案125中被进一步研究。一个建立在习题30基础上的计算机程序产生了下面的结果：

m	类	函数	m	类	函数	m	类	函数
0	1	1	5	87	17472	10	7	632
1	5	81	6	70	12696	11	1	96
2	18	1324	7	43	7408	12	2	24
3	46	6608	8	24	3346	13	1	16
4	87	14536	9	10	1296	14	0	0

下面是五元函数的相应统计数据：

m	类	函数	m	类	函数	m	类	函数
0	1	1	11	186447	666555696	22	338	608240
1	6	243	12	165460	590192224	23	130	197440
2	37	14516	13	129381	459299440	24	71	75720
3	244	318520	14	91026	319496560	25	37	28800
4	1527	3319580	15	57612	199792832	26	15	10560
5	6997	19627904	16	33590	113183894	27	6	2880
6	23434	73795768	17	17948	58653984	28	4	1040
7	57048	190814016	18	8880	27429320	29	2	640
8	105207	362973410	19	3986	11597760	30	2	48
9	152763	538238660	20	1795	4548568	31	2	64
10	183441	652555480	21	720	1633472	32	1	16

119. 有几位作者已经推测出 $b(n)=\hat{b}(n)$；M. M. Gadzhiev已经证明了当$n\leqslant 6$那个等式成立 [《Diskretnyĭ Analiz》18 (1971), 3—24]。

120. (a) 每个基本蕴涵项是一个极小项，因为n-立方体中没有相邻的点具有相同的奇偶性。所以在这种情况下满析取范式是唯一合适的DNF。

(b) 现在所有的基本蕴涵项由两个相邻的点构成。为了包含具有νx=1和νx=6的点，我们必须包括那14个子立方体$0^j{*}0^{6-j}$和$1^j{*}1^{6-j}$ $(0\leqslant j\leqslant 6)$。其他的$\binom{7}{3}+\binom{7}{4}=70$个点可以由35个精心挑选的基本蕴涵项包含（比如，参见6.5节习题1，或

者7.2.1.6节中的“圣诞树模型”)。这样最短DNF的长度为49。[S. B. Yablonsky在《Problemy Kibernetiki》7 (1962), 229—230中提出了一个巧妙的但似是而非的谬论说70个基本蕴涵项是必须的]。

(c) 对于$(x_1,\cdots,x_{n-1})$的2^{n-1}个选择中的每一个我们需要至多一个蕴涵项来解释函数关于x_n的行为。

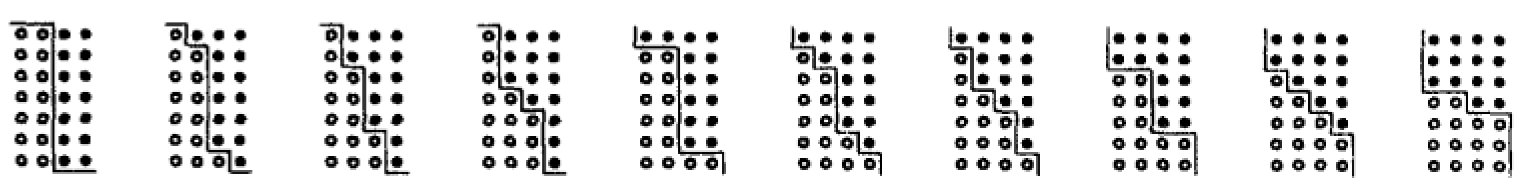
[渐近地，几乎所有的n元布尔函数有一个最短DNF，它有$\Theta(2^n/(\log n\log\log n))$个基本蕴涵项的。参见R. G. Nigmatullin,《Diskretnyĭ Analiz》10 (1967), 69—89; V. V. Glagolev,《Problemy Kibernetiki》19 (1967), 75—94; A. D. Korshunov,《Metody Diskretnogo Analiza》37 (1981), 9—41; N. Pippenger,《Random Structures & Algorithms》22 (2003), 161—186。]

121. (a) 令$x=x_1\cdots x_m$，$y=y_1\cdots y_n$。因为f是一个$(\nu x,\nu y)$函数，故共有$2^{(m+1)(n+1)}$种可能性。

(b) 在这种情况下$\nu x\leqslant\nu x'$且$\nu y\leqslant\nu y'$蕴涵$f(x,y)\leqslant f(x',y')$。每一个这样的函数对应于一条从$a_0=\left(-\frac{1}{2},n+\frac{1}{2}\right)$到$a_{m+n+2}=\left(m+\frac{1}{2},-\frac{1}{2}\right)$的交错路径，其中$a_j=a_{j-1}+(1,0)$或者$a_j=a_{j-1}-(0,1)$，$1\leqslant j\leqslant m+n+2$；我们有$f(x,y)=1$当且仅当点$(\nu x,\nu y)$落在路径上。所以可能性的数目就是这种路径的数目，也就是$\binom{m+n+2}{m+1}$。

(c) 对x和y取补使νx变为$m-\nu x$，νy变为$n-\nu y$。所以当m和n都是偶数时没有这样的函数；其他情况时则有$2^{(m+1)(n+1)/2}$个。

(d) (b)中的路径现在必须满足$a_j+a_{m+n+2-j}=(m,n)$，$0\leqslant j\leqslant m+n+2$。因此有$\binom{\lceil m/2\rceil+\lceil n/2\rceil}{\lceil m/2\rceil}$个 [$m$奇数或者$n$奇数]这样的函数。比如，当$m=3$，$n=6$时的10种情况是



122. 这种类型的函数是正则的且x'在y'的左边当且仅当那条交错路径不包含两个点(x,y)和$(x+2,y)$（其中$0<y<n$）；它是正则的且y'在x'的左边当且仅当那交错路径不包含$(x,y+2)$和(x,y)（其中$0<x<m$）。它是一个阈函数当且仅当有一条直线穿过点$(m/2,n/2)$且这条直线要具有性质：对于$0\leqslant s\leqslant m$，$0\leqslant t\leqslant n$，(s,t)在直线上方当且仅当(s,t)在路径上方。所以在前面答案中举例说明的情况5和8不是正则的；情况1,2,3,7,9和10是阈函数。剩下的正则非阈函数可以同样被如下表示：$((x_1\vee x_2\vee x_3)\wedge\langle x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle)\vee(x_1\wedge x_2\wedge x_3)$(情况4)；$\langle 00x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle\vee(\langle x_1x_2x_3\rangle\wedge\langle 11x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle)$ (情况6)。

123. 自对偶正则函数是相对容易列举的，但是数量增加地很快；当$n=9$时，有319 124个被Muroga、Tsuboi和Baugh于1967年发现。(当$n\leqslant 6$时，对应的数量出现

在表5中，因为所有这样的函数在$n<9$时都是阈函数。当$n=7$时有135个，当$n=8$时有2470个。）对于任何这样的函数，其阈值条件可以通过对习题103的方法改进而被迅速检验，因为只有极小向量x（关于众数）需要限制以使得$f(x)=1$。

我们已经知道了n元阈函数的θ_n值要满足$\lg\theta_n=n^2-O(n^2/\log n)$；参见Yu. A. Zuev，《Matematicheskie Voprosy Kibernetiki》5 (1994)，5—61。

124. 表5中列出的222个等价类包括24个大小为$2^{n+1}n!=768$的类；所以有$24\times768=18432$个问题的解答。其中一个是函数$(w\wedge(x\vee(y\wedge z)))\oplus z$。

125. 0；x；$x\wedge y$；$x\wedge y\wedge z$；$x\wedge(y\vee z)$；$x\wedge(y\oplus z)$。（这些函数是$x\wedge(f(y,z))$，其中f通过置换或者变量来运行二元函数的等价类，而不是通过函数值的取补来运行。一般地，令$f\simeq g$表示在那个较弱的意义上f等价于g，但是如果它们在表5的意义下是相等的，则写为$f\cong g$。因此$x\wedge f\cong x\wedge g$当且仅当$f\simeq g$，假定f和g对于变量x是独立的。因为容易看到$(x\wedge f)\simeq(\bar{x}\vee\bar{g})$是不可能的。并且如果$(x\wedge f)\simeq(x\wedge g)$，我们可以通过证明如果$\sigma$是$\{x_0,\cdots,x_n\}$的一个带正负号的置换并且如果$x=x_1\cdots x_n$，则恒等式$x_0\wedge f(x)=(x_0\sigma)\wedge g(x\sigma)$蕴涵$f(x)=g(x\sigma\tau)$来证明$f\simeq g$，其中$\tau$交换$x_0\leftrightarrow x_0\sigma$。从而表5的最下面一行枚举出了在$\simeq$下的等价类，但是同时$n$增加1；比如有402个这样的4元函数类。）

126. (a) 函数是渠限化的当且仅当它存在一个基本蕴涵项含有至多一个文字，或者一个基本子句含有至多一个文字。

(b) 函数是渠限化的当且仅当至少$\Sigma(f)$的一个成分等于0，2^{n-1}，$N(f)$或$N(f)-2^{n-1}$[参见I. Shmulevich、H. Lähdesmäki、K. Egiazarian，《IEEE Signal Processing Letters》11 (2004), 289—292, Proposition 6。]

(c) 如果说$\vee(f)=y_1\cdots y_n$, $y_j=0$，则只要$x_j=1$就有$f(x)=0$。因此f是渠限化的当且仅当我们没有$\vee(f)=\vee(\bar{f})=1\cdots1$和$\wedge(f)=\wedge(\bar{f})=0\cdots0$。通过这个检验，我们可以证明若函数的值只在几个点被知道，则它是非渠限化的。

127. (a) 由于一个自对偶函数$f(x_1,\cdots,x_n)$恰好在2^{n-1}个点为真，它关于变量x_j是渠限化的当且仅当$f(x_1,\cdots,x_n)=x_j$。

(b) 一个确定性Horn函数显然是渠限化的，如果（i）它包含任何只含一个文字的子句，或者（ii）某个文字出现在每个子句中。否则它不是渠限化的。因为我们有$f(0,\cdots,0)=f(1,\cdots,1)=1$，由于（i）是错的；并且如果$x_j$是任何一个变量，则有一个子句$C_0$不包含$\bar{x}_j$和一个子句$C_1$不包含$x_j$，由于（ii）是错的。通过适当选取其他变量的值，我们可以使得当$x_j=0$时，$C_0\wedge C_1$为假，并且当$x_j=1$时也是。

128. 比如$(x_1\wedge\cdots\wedge x_n)\vee(\bar{x}_1\wedge\cdots\wedge\bar{x}_n)$。

129. $\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}2^{2^{n-2}+k+1}-2(n-1)-4(n\bmod 2)=n2^{2^{n-1}+2}+O(n^2 2^{2^{n-2}})$。

[参见W. Just, I. Shmulevich和J. Konvalina，《Physica》D197 (2004), 211—221。]

130. (a) 如果有a_n个n（或更少）元函数，但是有b_n个恰好为n元的函数，我们有$a_n=\sum_k\binom{n}{k}b_k$。因此$b_n=\sum_k(-1)^{n-k}\binom{n}{k}a_k$。（这一规则由C. E. Shannon注解在

《Trans. Amer. Inst. Electrical Engineers》57 (1938), 713—723，§4中，适用于表3中的所有列，除了对称函数的情况。）特别地，这里寻求的答案是168−4·20+6·6−4·3+2=114。

(b) 如果有a'_n个实质上不同的n（或者更少）元函数，并且有b'_n个恰好为n元的函数，我们有$a'_n=\sum_{k=0}^{n}b'_k$。从而$b'_n=a'_n-a'_{n-1}$，并且在这种情形下的答案是30-10=20。

131. 令有$h(n)$个Horn函数和$k(n)$个Krom函数。显然$\lg h(n)\geqslant\binom{n}{\lfloor n/2\rfloor}$，$\lg k(n)\geqslant\binom{n}{2}$。V. B. Alekseyev [《Diskretnaiă Matematika》1 (1989), 129—136]已经证明了$\lg h(n)=\binom{n}{\lfloor n/2\rfloor}(1+O(n^{-1/4}\log n))$。B. Bollobas、G. Brightwell和I. Leader [《Israel J. Math.》133 (2003), 45—60] 已经证明了$\lg k(n)\sim\frac{1}{2}n^2$。

132. (a) 提示是对的，因为

$$\sum_y s(y)s(y\oplus z)=\sum_{w,x,y}(-1)^{f(w)+w\cdot y+f(x)+x\cdot(y+z)}=2^n\sum_{w,x}(-1)^{f(w)+f(x)+x\cdot z}[x=w]$$

现在假定对于x的$2^{n-1}+k$个值有$f(x)=g(x)$；则对于x的$2^{n-1}-k$个值有$f(x)=g(x)\oplus 1$。但是如果对于所有的仿射g有$|k|<2^{n/2-1}$，则对于所有的y我们将有$|s(y)|<2^{n/2}$，当$z=0$时与提示矛盾。

(b) 给定$y=y_1\cdots y_n$，对于$1\leqslant k\leqslant n/2$，当$x_{2k}=y_{2k-1}$时$f(x)=x\cdot y \bmod 2$恰有$2^{n/2}((y_1y_2+y_3y_4+\cdots+y_{n-1}y_n+1+h(y_1,y_3,\cdots,y_{n-1}))\bmod 2)$个解，并且对于$(x_2,x_4,\cdots,x_n)$的其他$2^{n/2}-1$个值，其每个都有$2^{n/2-1}$个解。所以总共有$2^{n-1}\pm 2^{n/2-1}$个解。（这个论证证明了，实际上，当$g(x_1,x_3,\cdots,x_{2n-1})$是所有的$2^{n/2}$位向量的一个排列时，$(g(x_1,x_3,\cdots,x_{2n-1})\cdot(x_2,x_4,\cdots,x_{2n})+h(x_2,x_4,\cdots,x_{2n})\bmod 2$是弯曲的。）

(c) 如果$f(x)$是弯曲的，(a) 中的论证证明了对于某个布尔函数$g(y)$有$s(y)=2^{n/2}(-1)^{g(y)}$。这个函数也是弯曲的，因为对于所有的w，$\sum_y(-1)^{g(y)+w\cdot y}=2^{-n/2}\sum_{x,y}(-1)^{f(x)+x\cdot y+w\cdot y}=2^{n/2}\sum_x(-1)^{f(x)}[x=w]=2^{n/2}(-1)^{f(w)}$。因此提示告诉我们对于所有的非零$z$有$\sum_x(-1)^{f(x)+f(x\oplus z)}=0$。

相反地，假定$f(x)$满足所说的条件，并且对于x的恰好k个值有$f(x)=1$。如果y是任何布尔向量，函数$g(x)=(f(x)+x\cdot y)\bmod 2$同样也满足条件，因此我们只需要证明$k=2^{n-1}\pm 2^{n/2-1}$。

令A为$2^n\times 2^n$矩阵，在行x和列y上为$f(x\oplus y)$，并且令J为所有1的$2^n\times 2^n$矩阵。那么我们有$J^2=2^nJ$和$JA=AJ=kJ$，从而

$$2^nI=(J-2A)^2=2^nJ-4kJ+4A^2$$

后面接着，当$y\neq 0\cdots 0$，使得$f(x)=1$并且$f(x\oplus y)=1$的x的数目为$k-2^{n-2}$。换句话说，恰

有$k-2^{n-2}$个向量有序对（w，x）使得$f(w)=f(x)=1$并且$w\oplus x=y$。对于所有的y求和，得出$(2^n-1)(k-2^{n-2})=k(k-1)$，并且这个二次方程的解为$k=2^{n-1}\pm 2^{n/2-1}$。

（d）由习题11，项$x_1\cdots x_r$存在当且仅当等式$f(x_1,\cdots,x_r,0,\cdots,0)=1$有奇数个解，并且一个等价条件是$\left(\sum_{x_1,\cdots,x_r}(-1)^{f(x_1,\cdots,x_r,0,\cdots,0)}\right)\bmod 4=2$。我们已经在（c）中看到这个和为 $2^{-n}\sum_{x_1,\cdots,x_r,y}s(y)(-1)^{x_1y_1+\cdots+x_ry_r}=2^{r-n}\sum_{y_{r+1},\cdots,y_n}s(0,\cdots,0,y_{r+1},\cdots,y_n)$。
如果$r=n$，后者和为$\pm 2^{n/2}$；其他情况它包含了被加数的一个偶数，其中的每一个是$\pm 2^{r-n/2}$。所以结果是4的倍数。

[弯曲函数是由O. S. Rothaus于1966年引进的，他的私人传阅文件（privately circulated paper）最终出版于《J. Combinatorial Theory》A20 (1976), 300—305. J. F. Dillon,《Congressus Numerantium》14 (1975), 237—249中，他发现了另外的弯曲函数的族，并且当$n\geqslant 8$且n是偶数时的其他许多例子后来也被相继发现。当n是奇数时弯曲函数不存在，但是当g和h是弯曲的时候，一个像$g(x_1,\cdots,x_{n-1})\oplus x_n\wedge h(x_1,\cdots,x_{n-1})$的函数与所有仿射函数的距离为$2^{n-1}-2^{(n/3)/2}$。对于情况$n=15$的一个更好的构造由N. J. Patterson和D. H. Wiedemann找到，《IEEE Transactions》IT-29 (1983), 354—356; IT-36(1990),443, 达到距离$2^{14}-108$。]

133. 令 $p_k=1/2(2^{n-k}+1)$，这样 $\bar{p}_k=2^{2^{n-k}}/(2^{2^{n-k}}+1)$。[Ph.D. thesis (MIT.1994)。]

7.1.2节

1. $((x_1 \vee x_4) \wedge x_2) \equiv (x_1 \vee x_3)$。

2. (a) $(w \oplus (x \wedge y)) \oplus ((x \oplus y) \wedge z)$；(b) $(w \wedge (x \vee y)) \wedge ((x \wedge y) \vee z)$。

3. [《Doklady Akademii Nauk SSSR》 115 (1957), 247—248。] 构建一个$k \times n$的矩阵，它的行是向量x，其中$f(x)=1$。通过排列和/或补充变量，我们可以假定头一行是$1\cdots1$，并且列都是排序过的。假设存在l个不同的列。那么$f=g \wedge h$，其中g是使得列$j-1$等于列j对于所有$1<j \leqslant n$成立的表达式$(x_{j-1} \equiv x_j)$的与，h是长度为l的k个最小项的或，它使用了来自每个包含相等列的组的一个变量。例如，如果$n=8$且f在$k=3$点11111111,00001111,00110111的值为1，那么$l=4$且 $f(x) = (x_1 \equiv x_2) \wedge (x_3 \equiv x_4) \wedge (x_6 \equiv x_7) \wedge (x_7 \equiv x_8) \wedge ((x_1 \wedge x_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge x_3 \wedge \bar{x}_5 \wedge x_6))$。这个表达式的长度一般是$2n+(k-2)l-1$，并且我们有$1 \leqslant 2^{k-1}$。

请注意k的值较大，我们通过将$f(x)$写成析取$f_1(x) \vee \cdots \vee f_r(x)$可以得到更短的表达式，其中每个$f_j$有最多$\lceil k/r \rceil$个1。因此

$$L(f) \leqslant \min_{r \geqslant 1}(r-1+(2n+\lceil k/r-2 \rceil 2^{\lceil k/r-1 \rceil})r)$$

4. 第一个不等式很明显，因为一棵深度为d的二叉树有最多$1+2+\cdots+2^{d-1}=2^d-1$个内部节点。

然后考虑提示，因为我们能够找到一棵规模$\geqslant \lfloor r/3 \rfloor$的最小子树。它的规模$s$最多是$1+2(\lfloor r/3 \rfloor - 1)$。因此我们能够得出$f=(g?f_1:f_0)$，其中$g$是一个规模为$s$的子表达式；$f_0$和$f_1$是那个子表达式分别由0和1替换而得到的长度为$r-s-1$的表达式。

设$d(r)=\max\{D(f) | L(f)=r\}$。因为多路复用函数的深度为2，且$\max(s,r-s-1)<\left\lceil \frac{2r}{3} \right\rceil$，对于$r \geqslant 3$我们有$d(r) \leqslant 2+ d\left(\left\lceil \frac{2r}{3} \right\rceil - 1\right)$，结果由$r$上的归纳可得 [《Hawaii International Conf. System Sci.》 4(1971), 525—527]。

5. 设$g_0=0$，$g_1=x_1$，对于$j \geqslant 2$有$g_j=x_j \wedge (x_{j-1} \vee g_{j-2})$。那么$F_n=g_n \vee g_{n-1}$，它的代价是$2n-2$，深度是$n$。[这些函数$g_j$也在二进制加法中起着突出的作用；参见习题42和44中以深度$O(\log n)$计算它们的方法。]

6. 正确：考虑$y=0$和$y=1$的情况。

7. $\hat{x}_5 = x_1 \vee x_4$，$\hat{x}_6 = x_2 \wedge \hat{x}_5$，$\hat{x}_7 = x_1 \vee x_3$，$\hat{x}_8 = \hat{x}_6 \oplus \hat{x}_7$。（原始的链计算“随机”函数(6)；参见习题1。这条新的链计算那个函数的范式，即它的补集。）

8. 正如(7)中所示，所需的真值表由2^{n-k}个0的模块和2^{n-k}个1的模块交替组成。因此，如果我们乘以$(2^{n-k}+1)$将得到$x_k+(x_k<<2^{n-k})$，所有的值都是1。

9. 当在步骤L6中找到$L(f)=\infty$时，我们能够将g和h存储在与f相关的一个记录里。然后一个递归的过程能够从g和h各自的表达式为f构建一个最短长度的表达式。

10. 在L3这一步，使用$k=r-1$而不是$k=r-1-j$。同理在所有地方将L改成D。

11. 唯一值得关注的一点是j应该在步骤U3时减小；那么当$j=0$时我们将不会有$\phi(g)\&\phi(h)\neq0$，因此在我们开始查看表$r-1$之前，所有代价为$r-1$的情况都能被

发现。

U1. [初始化。] 设$U(0)\leftarrow\phi(0)\leftarrow0$和$U(f)\leftarrow\infty$，其中$1\leqslant f<2^{2^n-1}$。然后设$U(x_k)\leftarrow\phi(x_k)\leftarrow0$，并将$x_k$放到表0中，如在步骤L1中一样。同时设$U(x_j\circ x_k)\leftarrow1$，设$\phi(x_j\circ x_k)$为权重为1的一个合适的位向量，并将$x_j\circ x_k$放到表1中，对于$1\leqslant j<k\leqslant n$和所有五种正规操作符$\circ$。最终设 $c\leftarrow2^{2^n-1}-5\binom{n}{2}-n-1$。

U2. [对r循环。] 当$c>0$时，对$r=2,3,\cdots$反复执行步骤U3。

U3. [对j和k循环。] 当$j\geqslant0$时，对 $j=\lfloor(r-1)/2\rfloor$，$\lfloor(r-1)/2\rfloor-1$，…，和$k=r-1-j$反复执行步骤U4。

U4. [对g和h循环。] 对表j中的所有g和表k中的所有h执行步骤U5；如果$j=k$，那么将h限制为表k中紧跟在g之后的函数。

U5. [对f循环。] 如果$\phi(g)\&\phi(h)\neq0$，那么设$u\leftarrow r-1$和$v\leftarrow\phi(g)\&\phi(h)$；否则设$u\leftarrow r$和$v\leftarrow\phi(g)|\phi(h)$。然后对$f=g\&h$，$f=\overline{g}\&h$，$f=g\&\overline{h}$，$f=g|h$和$f=g\oplus h$执行步骤U6。

U6. [更新$U(f)$和$\phi(f)$。] 如果$U(f)=\infty$，那么设$c\leftarrow c-1$，$\phi(f)\leftarrow v$，并将f放到列表u中。否则，如果$U(f)>u$，那么设$\phi(f)\leftarrow v$并将f从列表$U(f)$中移到列表u中。否则，如果$U(f)=u$，那么设$\phi(f)\leftarrow\phi(f)|v$。▌

12. $x_4=x_1\oplus x_2$，$x_5=x_3\wedge x_2$，$x_6=x_2\wedge\bar{x}_4$，$x_7=x_5\vee x_6$。

13. f_5=01010101(x_3)； $f_4=01110111(x_2\vee x_3)$； $f_3=01110101((\bar{x}_1\wedge x_2)\vee x_3)$；$f_2$=00110101$(x_1?x_3{:}x_2)$；$f_1$=00010111$(\langle x_1x_2x_3\rangle)$。

14. 对于$1\leqslant j\leqslant n$，首先计算$t\leftarrow(g\oplus(g>>2^{n-j}))\&x_j$，$t\leftarrow t\oplus(t<<2^{n-j})$，其中$x_j$是真值表(11)；那么对于$1\leqslant k\leqslant n$和$k\neq j$，所需要的与$x_j\leftarrow x_j\circ x_k$相关的真值表是$g\oplus(t\&((x_j\circ x_k)\oplus x_j))$。

($5n(n-1)$个掩码（mask）$(x_j\circ x_k)\oplus x_j$都独立于g，并且能够提前计算出来。如果我们允许形式是$x_{j(i)}\leftarrow x_{k(i)}\circ_i x_{l(i)}$的更一般的计算，同样的想法对于$5n^2(n-1)$个掩码$(x_k\circ x_l)\oplus x_j$也成立。)

15. 值得注意的是用非对称的方法来计算对称的函数：

(a)	(b)	(c)
$x_1\leftarrow x_1\oplus x_2$	$x_1\leftarrow x_1\oplus x_2$	$x_1\leftarrow x_1\oplus x_2$
$x_3\leftarrow x_3\oplus x_4$	$x_2\leftarrow x_2\wedge\bar{x}_1$	$x_2\leftarrow x_2\oplus x_3$
$x_1\leftarrow x_1\oplus x_3$	$x_3\leftarrow x_3\oplus x_4$	$x_2\leftarrow x_2\vee x_1$
$x_2\leftarrow x_2\oplus x_4$	$x_4\leftarrow x_4\wedge x_1$	$x_1\leftarrow x_1\oplus x_4$
$x_3\leftarrow x_3\vee x_2$	$x_2\leftarrow\bar{x}_2\wedge x_3$	$x_1\leftarrow x_1\wedge x_3$
$x_3\leftarrow x_3\wedge\bar{x}_1$	$x_2\leftarrow x_2\oplus x_1$	$x_2\leftarrow x_2\wedge\bar{x}_1$
	$x_2\leftarrow x_2\wedge\bar{x}_4$	$x_2\leftarrow x_2\oplus x_4$

16. 只使用$\oplus$和求补运算的计算只会生成仿射函数（affine function）（参见7.1.1节习题132）。假设$f(x)=f(x_1,\cdots,x_n)$是一个可以用最小存储空间计算的非仿射函数。那么$f(x)$对于一些非奇异$n\times n$的0和1矩阵A有形式$g(Ax+c)$，其中$g(y_1,y_2,\cdots,y_n)=g(y_1\wedge y_2,y_2,\cdots,y_n)$，$x$和$c$是列向量且向量操作以2为模执行；在这个表达式里，矩阵A和向量c与所有的操作$x_i\leftarrow x_i\oplus x_j$和/或出现在最近被执行的非仿射操作之后的坐标的排列和补充有关。我们将会利用$g(0,0,y_3,\cdots,y_n)=g(1,0,y_3,\cdots,y_n)$这个事实。

设α和β是A的头两行；同时设a和b是c的头两个元素。那么如果$Ax+c\equiv y$

(modulo 2)，我们有$y_1=y_2=0$当且仅当$\alpha \cdot x\equiv a$和$\beta \cdot x\equiv b$。共有2^{n-2}个向量x满足这个条件，并且对于所有这样的向量我们有$f(x)=f(x\oplus w)$，其中$Aw\equiv(1,0,\cdots,0)^{\mathrm{T}}$。

给定α, β, a, b和w，$\alpha\neq(0,\cdots,0)$，$\beta\neq(0,\cdots,0)$，$\alpha\neq\beta$和$\alpha \cdot w\equiv 1$ (modulo 2)，存在$2^{2^n-2^{n-2}}$个函数f有性质：当$\alpha \cdot x \bmod 2=a$和$\beta \cdot x \bmod 2=b$时，$f(x)=f(x\oplus w)$。因此以最小存储空间可计算的函数总数最多是2^{n+1}（对于仿射函数）加

$$(2^n-1)(2^n-2)2^2(2^{n-1})(2^{2^n-2^{n-2}})<2^{2^n-2^{n-2}+3n+1}$$

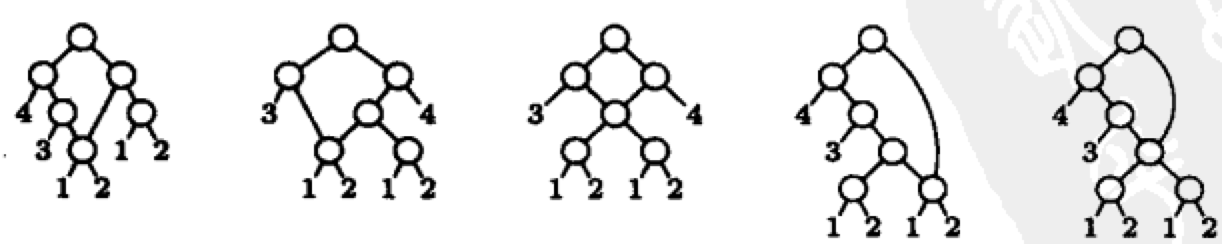
17. 设$f(x_1,\cdots,x_n)=g(x_1,\cdots,x_{n-1})\oplus(h(x_1,\cdots,x_{n-1})\wedge x_n)$，如7.1.1节(16)。以CNF表示$h$，在$x_0$中一个一个形成子句，并将它们做与运算得到$x_n$，就获得了$h\wedge x_n$。将$g$表示成为合取的和（对2取模），形成$x_0$上连续的合取，并当准备好时对它们做异或运算得到$x_n$。

（如果我们不允许使用非渠限化操作符$\oplus$和$\equiv$，在$n+1$个寄存器里计算所有的函数似乎是不可能的。但是$n+2$个寄存器肯定足够了，甚至限制为单一的操作符$\overline{\wedge}$时也行。）

18. 正如答案14中提到的那样，我们必须扩展正文中最小存储空间计算的定义，以便也能允许像$x_{j(i)}\leftarrow x_{k(i)}\circ_i x_{l(i)}$这样的步骤，其中$k(i)\neq j(i)$且$l(i)\neq j(i)$，因为那样能够为只依赖5个变量中的4个的特定函数给出更好的结果。那么我们分别为(2,2,5,20,93, 389,1960,10459,47604,135990,198092,123590,21540,472,0)类函数……找到$C_m(f)=(0,1,\cdots,13,14)$，为$C_m(f)=\infty$而剩下75 908个类（和575 963 136个函数），因为它们以最小存储空间完全不能计算。那一类中最有趣的函数可能是$(x_1\wedge x_2)\vee(x_2\wedge x_3)\vee(x_3\wedge x_4)\vee(x_4\wedge x_5)\vee(x_5\wedge x_1)$，它有$C(f)=7$但是$C_m(f)=\infty$。另一个有趣的例子是$(((x_1\vee x_2)\oplus x_3)\vee((x_2\vee\bar{x}_4)\wedge x_5))\wedge((x_1\equiv x_2)\vee x_3\vee x_4)$，它有$C(f)=8$和$C_m(f)=13$。在8步内计算出那个函数的一种方法是$x_6=x_1\vee x_2$，$x_7=x_1\vee x_4$，$x_8=x_2\oplus x_7$，$x_9=x_3\oplus x_6$，$x_{10}=x_4\oplus x_9$，$x_{11}=x_5\vee x_9$，$x_{12}=x_8\wedge x_{10}$，$x_{13}=x_{11}\wedge\bar{x}_{12}$。

19. 如果不是这样，那么根的左右子树必须重叠，因为情况(i)失败了。由假设可知，每个变量必须作为一个叶子节点出现至少一次。至少两个变量必须作为叶子节点出现至少两次，因为情况(ii)失败了。但是我们不能同时有$n+2$个叶子节点和$r\leqslant n+1$个内部节点，除非子树无法重叠。

20. 现在算法L（在步骤L5中省略了'$f=g\oplus h$'）表明一些表达式的长度必须是15；而且甚至习题11的足迹方法的效果实际上和14一样。为了得到真正的最短链，文中$r=6$的25条特殊链必须由5条不能再被排除的其他链补充，即



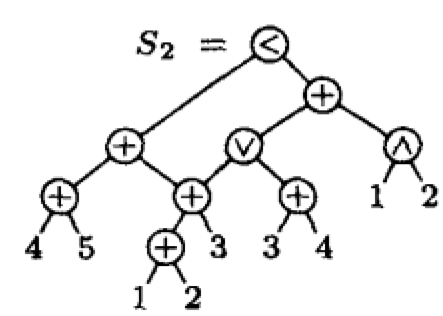
也就是当$r=(7,8,9)$时我们也必须分别考虑(653,12387,225660)条额外的潜在的链，它们不是从上至下和从下至上构造的特殊例子。下面是结果数据，可以与表1进行比较：

$C_c(f)$	类	函数	$U_c(f)$	类	函数	$L_c(f)$	类	函数	$D_c(f)$	类	函数
0	2	10	0	2	10	0	2	10	0	2	10
1	1	48	1	1	48	1	1	48	1	1	48
2	2	256	2	2	256	2	2	256	2	7	684
3	7	940	3	7	940	3	7	940	3	59	17064
4	9	2336	4	9	2336	4	7	2048	4	151	47634
5	24	6464	5	21	6112	5	20	5248	5	2	96
6	30	10616	6	28	9664	6	23	8672	6	0	0
7	61	18984	7	45	15128	7	37	11768	7	0	0
8	45	17680	8	40	14296	8	27	10592	8	0	0
9	37	7882	9	23	8568	9	33	11536	9	0	0
10	4	320	10	28	5920	10	16	5472	10	0	0
11	0	0	11	6	1504	11	30	6304	11	0	0
12	0	0	12	5	576	12	3	960	12	0	0
13	0	0	13	3	144	13	8	1472	13	0	0
14	0	0	14	2	34	14	2	96	14	0	0
15	0	0	15	0	0	15	4	114	15	0	0

两个深度为5的函数类由$S_{2,4}(x_1,x_2,x_3,x_4)$和$x_1\oplus S_2(x_2,x_3,x_4)$表示；而且那两个函数和$S_2(x_1,x_2,x_3,x_4)$，以及奇偶校验函数$S_{1,3}(x_1,x_2,x_3,x_4)=x_1\oplus x_2\oplus x_3\oplus x_4$的长度为15。同时$U_c(S_{2,4})=U_c(S_{1,3})=14$。代价为10的这四类由$S_{1,4}(x_1,x_2,x_3,x_4)$，$S_{2,4}(x_1,x_2,x_3,x_4)$，$(x_4?x_1\oplus x_2\oplus x_3:\langle x_1x_2x_3\rangle)$和$[(x_1x_2x_3x_4)_2\in\{0,1,4,7,10,13\}]$表示。（这些中的第三个刚好是(20)“哈佛最难的例子”的补充。）

21.（作者宣称他们的表入口“应仅被认为是当前作者所知道的最经济的操作符”。）

22. $\nu(x_1x_2x_3x_4x_5)=3$当且仅当$\nu(x_1x_2x_3x_4)\in\{2,3\}$且$\nu(x_1x_2x_3x_4x_5)$为奇数。相似地，$S_2(x_1,x_2,x_3,x_4,x_5)=S_3(\bar{x}_1,\bar{x}_2,\bar{x}_3,\bar{x}_4,\bar{x}_5)$包含$S_{1,2}(x_1,x_2,x_3,x_4)$：



23. 我们只需考虑32个规范的例子，如图9中，因为一个对称函数的补也是对称的。因此我们能够使用反射，如$S_{1,2}(x)=S_{3,4}(\bar{x})$，可能还有求补运算，如$S_{2,3,4,5}(x)=\overline{S}_{0,1}(x)=\overline{S}_{4,5}(\bar{x})$，来推导出绝大多数剩下的情况。当然通常情况下$S_1$，$S_{1,3,5}$和$S_{1,2,3,4,5}$的代价都是4。那样只剩下$S_{1,2,3,4}(x_1,x_2,x_3,x_4,x_5)=(x_1\oplus x_2)\vee(x_2\oplus x_3)\vee(x_3\oplus x_4)\vee(x_4\oplus x_5)$，在习题79中我们讨论过它对于一般的$n$的情况。

24. 正如文中提到的那样，猜想对于$n\leqslant 5$都有效。

25. 它的值是$2^{2^{n-1}}-n-1$，即非平凡正规函数的数量。（在任何不全包含这些函数的长度为r的正规链中，对于范围$1\leqslant j$，$k\leqslant n+r$内的一些j和k，以及一些正规二元操作符$\circ$，$x_j\circ x_k$会是一个新的函数；因此我们能够在每个新的步骤计算一个新的函数，直到我们全部得到它们。）

26. 错误。例如，如果$g=S_{1,3}(x_1,x_2,x_3)$且$h=S_{2,3}(x_1,x_2,x_3)$，那么$C(gh)=5$是一个完全加法器的代价，但是以图9来说，$f=S_{2,3}(x_0,x_1,x_2,x_3)$的代价为6。

27. 是的：操作‘$x_2\leftarrow x_1\oplus x_1$, $x_1\leftarrow x_1\oplus x_3$, $x_1\leftarrow x_1\wedge\bar{x}_2$, $x_1\leftarrow x_1\oplus x_3$, $x_2\leftarrow x_2\oplus x_3$’将$(x_1$,

x_2, x_3)转换成(z_1, z_0, x_3)。

28. 设$v'=v''=v\oplus(x\oplus y)$；$u'=((v\oplus y)\,\overline{\subset}\,(x\oplus y))\oplus u$，$u''=((v\oplus y)\vee(x\oplus y))\oplus u$。这样我们能在$j$是奇数时设$u_0=0$，$v_0=x_1$和$u_j=((v_{j-1}\oplus x_{2j+1})\vee(x_{2j}\oplus x_{2j+1}))\oplus u_{j-1}$；当$j$是偶数时设$u_j=((v_{j-1}\oplus x_{2j+1})\,\overline{\subset}\,(x_{2j}\oplus x_{2j+1}))\oplus u_{j-1}$；以及可以设$v_j=v_{j-1}\oplus(x_{2j}\oplus x_{2j+1})$，从而对于$1\leqslant j\leqslant \lfloor n/2\rfloor$得到$(u_jv_j)_2=(x_1+\cdots x_{2j+1})\bmod 4$。当$n$是偶数时设$x_{n+1}=0$。从而函数$[(x_1+\cdots+x_n)\bmod 4=0]=\bar{u}_{\lfloor n/2\rfloor}\wedge\bar{v}_{\lfloor n/2\rfloor}$能在$\lfloor 5n/2\rfloor-2$步内计算出来。

这个构建应归功于L. J. Stockmeyer，他证明了这种方法几乎是最优的。事实上，习题80的结果与图9和图10一起可以表明，对所有的$n\geqslant 5$，这种方法构建的链最多比一条最优的可能链长一步。

顺便说一句，由类似的式子$u'''=((v\oplus y))\wedge(x\oplus y))\oplus u$可推出$(u'''v')_2=((uv)_2+x-y)\bmod 4$。看起来更简单的函数$((uv)_2+x+y)\bmod 4$的代价是6，而不是5。

29. 为了得到一个上界，假定每个完全加法器和半加法器将深度增加3。如果存在权重为2^j和深度为$3d$的a_{jd}位，那么我们最多调度权重为$\{2^j, 2^{j+1}\}$和深度为$3(d+1)$的随后$\lceil a_{jd}/3\rceil$位。接着有归纳$a_{jd}\leqslant\binom{d}{j}3^{-d}n+4$。因此当$d\geqslant\log_{3/2}n$时$a_{jd}\leqslant 5$，且总深度最多是$3\log_{3/2}n+3$。（奇怪的是，当$n=10^7$时，真正的深度实际上刚好是100。）

30. 像通常一样，设νn代表n本身以二进制表示的位的旁路加法。那么$s(n)=5n-2\nu n-3\lfloor\lg n\rfloor-3$。

31. 在旁路加法以$s(n)<5n$步进行之后，由定理L知：一个$(z_{\lfloor\lg n\rfloor},\cdots,z_0)$的任意函数能够以最多$\sim 2n/\lg n$步计算出来。[参见O. B. Lupanov，《Doklady Akademii Nauk SSSR》140 (1961), 322—325。]

32. 引导程序：首先通过对n的归纳证明$t(n)\leqslant 2^{n+1}$。

33. 错误，在一个细节上有误：如果$N=\sqrt{n}$，至少需要n步。然而，一个正确的渐进的表达式$N+O(\sqrt{N})+O(n)$能够得到证明，方法是通过首先认识到正文中的方法在$N\geqslant 2^{n-1}$时给出$N+O(\sqrt{N})$；否则，如果$\lfloor\lg N\rfloor=n-k-1$，那么我们能够使用$O(n)$个操作来对量$\bar{x}_1\wedge\cdots\wedge\bar{x}_k$和其他的变量$x_{k+1},\cdots,x_n$做与运算，然后以$n-k$继续进行。

（一个结论是我们能够以代价$s(n)+n+O(\sqrt{n})=6n+O(\sqrt{n})$和深度$O(\log n)$来计算对称函数$\{S_1,S_2,\cdots,S_n\}$。）

34. 假设一个扩展的优先级编码器有$n+1=2^m$个输入$x_0x_1\cdots x_n$和$m+1$个输出$y_0y_1\cdots y_m$，其中$y_0=x_0\vee x_1\vee\cdots\vee x_n$。如果$Q'_m$和$Q''_m$是对$x'_0\cdots x'_n$和$x''_0\cdots x''_n$的扩展编码器，那么若我们定义了$y_0=y'_0\vee y''_0$, $y_1=y''_0$, $y_2=y_1?y''_1:y'_1,\cdots,y_{m+1}=y_1?y''_m:y'_m$，则$Q_{m+1}$也对$x'_0\cdots x'_nx''_0\cdots x''_n$起作用。如果$P'_m$是一个对$x'_1\cdots x'_n$的普通的优先级编码器，那么我们可以以相似的方式对$x'_1\cdots x'_nx''_0\cdots x''_n$得到$P_{m+1}$。

以$m=2$和$y_2=x_3\vee(x_1\wedge\bar{x}_2)$, $y_1=x_2\vee x_3$, $y_0=x_0\vee x_1\vee y_1$开始，这个构建生成代价分别为p_m和q_m的P_m和Q_m，其中$p_2=3$, $q_2=5$，且对于$m\geqslant 2$有$p_{m+1}=3m+p_m+q_m$, $q_{m+1}=3m+1+2q_m$。

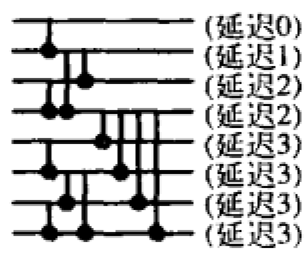
结果是$p_m=q_m-m$和$q_m=15\cdot 2^{m-2}-3m-4\approx 3.75n$。

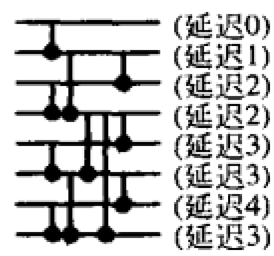
35. 如果$n=2m$，则计算$x_1\wedge x_2,\cdots,x_{n-1}\wedge x_n$，然后递归地产生$x_1\wedge\cdots\wedge x_{2k-2}\wedge x_{2k+1}\wedge\cdots\wedge x_n$，其中$1\leqslant k\leqslant m$，并在继续进行$n$步后结束。如果$n=2m-1$，那么对于$n+1$个元素使用这条链；通过设置$x_{n+1}\leftarrow 1$能够减少三个步骤。[I. Wegener, 《The Complexity of Boolean Functions》(1987)，习题3.25。同样的想法能够应用到任何联合和可换操作符来替代∧的情形。]

36. 按如下方法递归构建$P_n(x_1,\cdots,x_n)$和$Q_n(x_1,\cdots,x_n)$，其中P_n有最优的深度，Q_n的深度≤$\lceil\lg n\rceil+1$：$n=1$的情况是平凡的；否则P_n由$Q'_r(x_1,\cdots,x_r)$和$P''_s(x_{r+1},\cdots,x_n)$通过设$y_j=y'_j\,(1\leqslant j\leqslant r)$和$y_j=y'_r\wedge y''_{j-r}(r<j\leqslant n)$得到，其中$r=\lceil n/2\rceil$且$s=\lfloor n/2\rfloor$。而且$Q_n$是通过设$y_1=x_1$，$y_{2j}=y'_j$，$y_{2j+1}=y'_j\wedge x_{2j+1}(1\leqslant j\leqslant s)$和$y_{2s}=y'_s$，$y_n=y'_r$从$P'_r(x_1\wedge x_2,\cdots,x_{n-1}\wedge x_n)$或者$P'_r(x_1\wedge x_2,\cdots,x_{n-2}\wedge x_{n-1},\ x_n)$得到。

为了证明正确性，我们必须同时证明Q_n的输出y_n的深度是$\lceil\lg n\rceil$；请注意如果我们以$P'_m(x_1\wedge x_2,\cdots,x_{2m-2}\wedge x_m)$开始，而不是$P'_{m+1}(x_1\wedge x_2,\cdots,x_{2m-1}\wedge x_{2m},x_{2m+1})$，那么$Q_{2m+1}$将会失败，但是例外情况是当$m$为2的幂时。

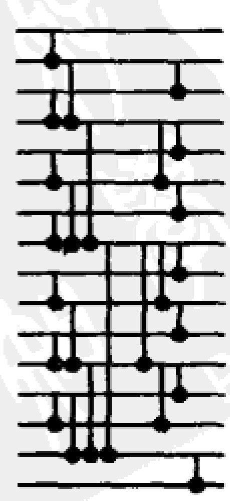
通过在第i步对一些指数$j(i)<k(i)$设$x_{k(i)}\leftarrow x_{j(i)}\wedge x_{k(i)}$，这些计算能够以最小内存运行。因此我们能够用类似于排序网络图的图来描述构建。例如，

P_8 = 

Q_8 = 

当$n>1$时，代价p_n和q_n满足 $p_n=\lfloor n/2\rfloor+q_{\lceil n/2\rceil}+p_{\lfloor n/2\rfloor}$， $q_n=2\lfloor n/2\rfloor-1+p_{\lceil n/2\rceil}$；例如，$(p_1,\cdots,p_7)=(q_1,\cdots,q_7)=(0,1,2,4,5,7,9)$。通过设$\overline{p}_n=4n-p_n$和$\overline{q}_n=3n-q_n$会生成更简单的表达式，这证明了$p_n<4n$和$q_n<3n$: $\overline{q}_n=\overline{p}_{\lceil n/2\rceil}+[n偶数]$; $\overline{p}_{4n}=\overline{p}_{2n}+\overline{p}_n+1$，$\overline{p}_{4n+1}=\overline{p}_{2n}+\overline{p}_{n+1}+1$，$\overline{p}_{4n+2}=\overline{p}_{2n+1}+\overline{p}_{n+1}$，$\overline{p}_{4n+3}=\overline{p}_{4n+2}+2$。特别地，$1+\overline{p}_{2m}=F_{m+5}$是一个斐波那契数。[参见《JACM》27(1980), 831—834。如果我们为Q_n在$n=2^m+1$时使用在其他情况下行不通的$P'_{\lfloor n/2\rfloor}$构造，如果我们用Q_5和Q_6来替代P_5和P_6，且如果我们进一步以$(Q_9,Q_{10},Q_{11},Q_{17})$来替代$(P_9,P_{10},P_{11},P_{17})$，那么会得到稍微改进的链。

请注意如果我们用任意的联合操作符来替代“∧”，那么这种构造可以在一般情况下有效。特别地，前缀的序列$x_1\oplus\cdots\oplus x_k(1\leqslant k\leqslant n)$定义了从Gray二进制码到以2为基的（radix-2）整数的转换，见7.2.1.1节等式(10)。



37. $m=15$, $n=16$的情况如右图描述。

(a) 设$x_{i..j}$代表$x_i\wedge\cdots\wedge x_j$的初始值。当这个算法设$x_k\leftarrow x_j\wedge x_k$时，可证明$x_k$的前一个值是$x_{j+1..k}$。在步骤S1之后，$x_k$成为$x_{f(k)+1..k}$，其中$f(k)=k\&(k-1)(1\leqslant k<m)$，$f(m)=0$。在步骤S2之后，$x_k$成为了$x_{1..k}$，其中$1\leqslant k\leqslant m$。

(b) S1的代价是$m-1$，S2的代价是 $m-1-\lceil \lg m \rceil$，S3的代价是$n-m$。x_k的最终延迟是 $\lceil \lg k \rceil+\nu k-1(1\leqslant k<m)$，$\lceil \lg m \rceil+k-m(m\leqslant k\leqslant n)$。因此$\{x_1,\cdots,x_{m-1}\}$的最大延迟是$g(m)=m-1(m<4)$，$g(m)=\lfloor \lg m \rfloor+\left\lfloor \lg \frac{m}{3} \right\rfloor(m\geqslant 4)$。我们有$c(m,n)=m+n-2-\lceil \lg m \rceil$，$d(m,n)=\max(g(m), \lceil \lg m \rceil+n-m)$。因此当$n\geqslant m+g(m)-\lceil \lg m \rceil$时有$c(m,n)+d(m,n)=2n-2$。

(c) 一个值表揭示了当$n<8$时，$d(n)=\lceil \lg n \rceil$；当$n\geqslant 8$时，$d(n)=\lfloor \lg(n-\lfloor \lg n \rfloor+3) \rfloor+\left\lfloor \lg \frac{2}{3}(n-\lfloor \lg n \rfloor+3) \right\rfloor-1$。以另一种方式开始，我们有$d(n)>d(n-1)>0$当且仅当对于一些$k>1$有$n=2^k+k-3$或者$2^k+2^{k-1}+k-3$。当$n<8$时，最小的情况在$m=n$时出现；否则，它在 $m=n-\left\lfloor \frac{2}{3}(n-\lfloor \lg n \rfloor+3) \right\rfloor+2-[$对于一些$k$, $n=2^k+k-3]$时出现。

(d) 设$m\leftarrow m(n,d)$，其中$m(n,d(n))$在前一个语句中定义，且当$d>d(n)$时有$m(n,d)=m(n-1,d-1)$。[参见J. Algorithms 7(1986), 185—201。]

38. (a)从顶至底来说，$f_k(x_1,\cdots,x_n)$是一个初级的对称函数，它也被称作阈函数$S_{\geqslant k}(x_1,\cdots,x_n)$。(参见5.3.4节习题28，7.1.1节等式(90)。)

(b) 正如习题33的答案所示，在以≈$6n$步计算出$\{S_1,\cdots,S_n\}$后，我们能够应用习题37的方法继续进行$2n$步后结束。

但是更为有趣的是专门为2^m+1个阈函数$g_k(x_1,\cdots,x_m)=[(x_1\cdots x_m)_2\geqslant k](0\leqslant k\leqslant 2^m)$的计算设计一条布尔链。因为$[(x'x'')_2\geqslant(y'y'')_2]=[(x')_2\geqslant(y')_2+1]\vee([(x')_2\geqslant(y')_2]\wedge[(x'')_2\geqslant(y'')_2])$，所以一个与二进制解码器相似的分治构造可以以最多$2t(m)$的代价来解决这个问题。

进一步，如果$2^{m-1}\leqslant n<2^m$，那么用这种方法计算$\{g_1,\cdots,g_n\}$的代价$u(n)$是$2n+O(\sqrt{n})$，且当n较小时十分合理。

$$n=1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20$$
$$u(n)=0\ 1\ 2\ 4\ 7\ 7\ 8\ 12\ 15\ 17\ 19\ 19\ 20\ 21\ 22\ 27\ 32\ 34\ 36\ 36$$

以旁路加法开始，我们能够以$s(n)+u(n)\approx 7n$步排序n个布尔值。一个代价为$2\hat{S}(n)$的排序网络在当$n=4$时表现得更好，但当$n\geqslant 8$时则不尽如人意。[参见5.3.4节(11); D. E. Muller和F. P. Preparata,《JACM》22 (1975), 195—201。]

39. [《IEEE Transactions》C-29 (1980), 737—738。]相等关系

$$M_{r+s}(x_1,\cdots,x_r,x_{r+1},\cdots,x_{r+s};\ y_0,\cdots,y_{2^{r+s}-1})=M_r(x_1,\cdots,x_r;\ y'_0,\cdots,y'_{2^r-1})$$

其中 $y'_j=\bigvee_{k=0}^{2^s-1}(d_k\wedge y_{2^s j+k})$，$d_k$是一个应用到$(x_{r+1},\cdots,x_{r+s})$的$s$到$2^s$解码器的第$k$个输出，表明$C(M_{r+s})\leqslant C(M_r)+2^{r+s}+2^r(2^s-1)+t(s)$，其中$t(s)$是解码器的代价(30)。深度

是$D(M_{r+s})=\max(D_x(M_{r+s}),D_y(M_{r+s}))$，其中$D_x$和$D_y$分别代表$x$和$y$变量的最大深度；我们有$D_x(M_{r+s})\leqslant\max(D_x(M_r), 1+s+\lceil\lg s\rceil+D_y(M_r))$以及$D_y(M_{r+s})\leqslant 1+s+D_y(M_r)$。

取 $r=\lceil m/2\rceil$，$s=\lfloor m/2\rfloor$，则有$C(M_m)\leqslant 2^{m+1}+O(2^{m/2})$，$D_y(M_m)\leqslant m+1+\lceil\lg m\rceil$，$D_x(M_m)\leqslant D_y(M_m)+\lceil\lg m\rceil$。

40. 举个例子，我们可以设 $f_{nk}(x)=\bigvee_{j=1}^{n+1-k}(l_j(x)\wedge r_{j+k-1}(x))$，其中

$$l_j(x)=\begin{cases}x_j, & \text{如果}j \bmod k=0\\ x_j\wedge l_{j+1}(x) & \text{如果}j \bmod k\neq 0\end{cases}\quad \text{对于}1\leqslant j\leqslant n-(n \bmod k)$$

$$r_j(x)=\begin{cases}1, & \text{如果}j \bmod k=0\\ x_j\wedge x_{j-1}(x) & \text{如果}j \bmod k\neq 0\end{cases}\quad \text{对于}k\leqslant j\leqslant n$$

代价是 $4n-3k-3\left\lfloor\frac{n}{k}\right\rfloor-\left\lfloor\frac{n-1}{k}\right\rfloor+2-(n \bmod k)$。

当n较小或k较小时，一个递归的解决方案更好：观察到

$$f_{nk}(x)=\begin{cases}x_{n-k+1}\wedge\cdots\wedge x_k\wedge\\ \quad f_{(2n-2k)(n-k)}(x_1,\cdots,x_{n-k},x_{k+1},\cdots,x_n) & \text{对于}k<n<2k\\ f_{\lfloor(n+k)/2\rfloor k}(x_1,\cdots,x_{\lfloor(n+k)/2\rfloor})\vee\\ \quad f_{\lfloor(n+k-1)/2\rfloor k}(x_{\lfloor(n-k)/2\rfloor+1},\cdots,x_n) & \text{对于}n\geqslant 2k\end{cases}$$

这个解答的代价可以证明等于 $n-1+\sum_{j=1}^{n-k}\lfloor\lg j\rfloor(k\leqslant n<2k)$，且渐进地位于$(m+\alpha_k-1)n+O(km)$和$(m+2-2/\alpha_k)n+O(km)$之间(当$n\to\infty$)，其中 $m=\lfloor\lg k\rfloor$，$1<\alpha_k=(k+1)/2^m\leqslant 2$。

然而，综合采用这些方法效果更好；最优情况的代价未知。

41. 设$c(m)$是用“条件和”（conditional-sum）方法来计算$(x)_2+(y)_2$和$(x)_2+(y)_2+1$的代价，其中x和y有$n=2^m$位；设$c'(m)$是仅计算$(x)_2+(y)_2$这个更简单问题的代价。那么$c(m+1)=2c(m)+6\cdot 2^m+2$，$c'(m+1)=c(m)+c'(m)+3\cdot 2^m+1$。（和的第$z_n$位的代价是1；但是$n<k\leqslant 2n+1$时，第$z_k$位的代价是3，因为它们的形式是$c?a_k{:}b_k$，其中$c$是一个进位。）如果我们以$n=1$和$c(0)=3$，$c'(0)=2$开始，那么答案是$c(m)=(3m+5)2^m-2$，$c'(m)=(3m+2)2^m-m$。但是对于情况$n=2$的改进的构造允许我们以$c(1)=11$和$c'(1)=7$开始；那么答案是 $c(m)=\left(3m+\frac{7}{2}\right)2^m-2$，$c'(m)=\left(3m+\frac{1}{2}\right)2^m-m+1$。在每个情况下深度都是$2m+1$。[参见J. Sklansky,《IRE Transactions》EC-9 (1960), 226—231。]

42. (a)因为$\langle x_ky_kc_k\rangle=u_k\vee(v_k\wedge c_k)$，所以我们可以利用(26)和归纳法。

(b) 请注意 $U_k^{k+1}=u_k$和 $V_k^{k+1}=v_k$；在$j-i$上使用归纳。[参见A. Weinberger和J. L. Smith,《IRE Transactions》EC-5 (1956), 65—73；R. P. Brent和H. T. Kung,《IEEE Transactions》C-31 (1982), 260—264。]

(c) 首先，对于l=1,2,⋯,m−1，1⩽k⩽n和$h(l)$的所有倍数i计算V_i^k，其中$k_l \geqslant i \geqslant k_{l+1}$，$k_l = h(l)\lfloor (k-1)/h(l) \rfloor$代表$h(l)$的小于$k$的最大倍数。例如，当$l$=2，$k$=99时，我们计算 V_{96}^{99}，$V_{88}^{99} = V_{96}^{99} \wedge V_{88}^{96}$，$V_{80}^{99} = V_{88}^{99} \wedge V_{80}^{88}, \cdots, V_{64}^{99} = V_{72}^{99} \wedge V_{64}^{72}$；这是一个前置计算，它使用当$l$=2时计算好的值 $V_{96}^{99}, V_{88}^{96}, V_{80}^{88}, \cdots, V_{64}^{72}$。通过使用习题36的方法，步骤$l$对深度增加最多$l$级，它需要总共 $(p_1 + p_2 + \cdots + p_{2^l})n/2^l = O(2^l n)$ 个门。

然后，再一次对l=1,2,⋯,m−1和1⩽k⩽n使用“非滚动”式子

$$U_{k_{l+1}}^k = U_{k_l}^k \vee \bigvee_{\substack{k_l > j \geqslant k_{l+1} \\ h(l) \backslash j}} (V_{j+h(l)}^k \wedge U_j^{j+h(l)})$$

计算U_i^k($i=k_{l+1}$)。

例如，当l=3和k=99时，“非滚动”式子是

$$V_{64}^{99} = U_{96}^{99} \vee (V_{96}^{99} \wedge U_{88}^{96}) \vee (V_{88}^{99} \wedge U_{80}^{88}) \vee (V_{80}^{99} \wedge U_{72}^{80}) \vee (V_{72}^{99} \wedge U_{64}^{72})$$

每个这样的U_i^k都是最多2^l项的并，因此它可以以深度⩽l加上每项的深度计算出来。对于1⩽k⩽n，这个阶段的总代价是$(0+2+4+\cdots+(2^l-2))n/2^l=O(2^l n)$。

因此，计算所有必要的U和V的总代价是 $\sum_{l=1}^{m-1} O(2^l n) = O(2^m n)$ 。（进一步说，分量V_0^k实际上并不需要，因此我们节省了 $\sum_{l=1}^{m-1} h(l) p_{2^l}$ 个门的代价。）例如，当m=(2,3,4,5)时，我们分别为(2,8,64,1024)位数字的加法得到布尔链，总深度分别是(3,7,11,16)，总代价分别是(7,64,1254,48470)。

[这个构造应归功于V. M. Khrapchenko，《Problemy Kibernetiki》 19 (1967), 107—122，他还展示了如何将它与别的方法联合起来，使得总代价是$O(n)$，同时深度还是 $\lg n + O(\sqrt{\log n})$。然而，他的联合方法只有纯理论上的趣味性，因为它要求在深度变得小于$2\lg n$之前有$n>2^{64}$。另一种在(b)中利用循环获得较小深度的方法可以是基于斐波那契数：斐波那契方法以深度$\log_\phi n+O(1)\approx 1.44\lg n$和代价$O(n \log n)$来计算进位。例如，它为二进制加法生成具有如下特征的链。

n =	4	8	16	32	64	128	256	512	1024
深度	6	7	9	10	12	13	15	16	18
代价	24	71	186	467	1125	2648	6102	13775	30861

参见D. E. Knuth, 《The Stanford GraphBase》 (1994), 276—279。

Charles Babbage为类似的基数为10的问题找到了一个天才而巧妙的方法，他的方法能够在固定的时间内对任意精度的数进行加法运算；为了使其能够正常工作，他需要理想化的、有着消失清除功能的刚性部件。参见H. P. Babbage, 《Babbage's Calculating Engines》 (1889), 334—335。一个等价的想法在物理晶体管上工作得很好，尽管它不能以布尔链的方式来表达；参见P. M. Fenwick, 《Comp.J.》 30 (1987), 77—79。]

43. (a)设$A=B=Q=\{0,1\}$，$q_0=0$。定义 $c(q,a) = d(q,a) = \bar{q} \wedge a$。

(b)关键的想法是构造函数$d_1(q)\cdots d_{n-1}(q)$，其中$d_1(q)=d(q,a_1)$和$d_{j+1}(q)=d(d_j(q),a_j)$。

换句话说，$d_1=d^{(a_1)}$，$d_{j+1}=d_j\circ d^{(a_j)}$，其中$d^{(a)}$是取 $q\mapsto d(q,a)$ 的函数，$\circ$代表函数的复合。每个函数d_j都能够以二进制表示法来编码，并且$\circ$是一个在这些二进制表示上的联合操作。因此函数$d_1d_2\cdots d_{n-1}$是前缀$d^{(a_1)}$，$d^{(a_1)}\circ d^{(a_2)}$，…，$d^{(a_1)}\circ\cdots\circ d^{(a_{n-1})}$；而且$q_1q_2\cdots q_n=q_0d_1(q_0)\cdots d_{n-1}(q_0)$。

(c)将一个函数$f(q)$用它的真值表f_0f_1来代替。那么组合$f_0f_1\circ g_0g_1$会是h_0h_1，其中函数$h_0=f_0?g_1:g_0$和$h_1=f_1?g_1:g_0$是都能以代价3和深度2计算出来的复用器。（联合的代价$C(h_0h_1)$的值仅为5，但是我们尝试将深度保持为一个小值。）$d^{(a)}$的真值表是$a0$。通过使用习题36，我们就能以代价$\leqslant 6p_{n-1}<24n$和深度$\leqslant 2\lceil\lg(n-1)\rceil$计算真值表$d_{10}d_{11}d_{20}d_{21}\cdots d_{(n-1)0}d_{(n-1)1}$；那么$b_j=\bar{q}_j\wedge a_j=\bar{d}_{(j-1)0}\wedge a_j$。（这些代价的估计是相当保守的；由于在$d^{(a_j)}$的初始真值表中的0，以及中间值$d_{j1}$中的许多都从未使用过，会产生很多简化。例如，当$n=5$时真正的代价只有10，而不是$6p_4+4=28$；真正的深度是4，而不是$1+2\lceil\lg 4\rceil=5$。）

44. 输入可以看作是字符串$x_0y_0x_1y_1\cdots x_{n-1}y_{n-1}$，它的元素属于4字母的字母表$A=\{00, 01, 10, 11\}$；存在两种状态$Q=\{0,1\}$代表一个可能的进位，其中$q_0=0$；输出的字母表是$B=\{0,1\}$；同时我们有$c(q,xy)=q\oplus x\oplus y$，$d(q,xy)=\langle qxy\rangle$。因此，在这种情况下，有限状态的变换器实质上是用一个全加器来描述。

当我们计算映射$d^{(xy)}$的时候，4个可能的q的函数中只有3个会出现。我们可以将它们编码成$u\vee(q\wedge v)$。初始的函数$d^{(xy)}$有$u=x\wedge y$，$v=x\oplus y$；同时组合$(uv)\circ(u'v')$是$u''v''$，其中$u''=u'\vee(v'\wedge u)$，$v''=v\wedge v'$。

例如，当$n=4$时，链有如下的形式，它使用了习题42的表示法：$U_k^{k+1}=x_k\wedge y_k$，$V_k^{k+1}=x_k\oplus y_k$，其中$0\leqslant k<4$；$U_0^2=U_1^2\vee(V_1^2\wedge U_0^1)$，$U_2^4=U_3^4\vee(V_3^4\wedge U_2^3)$，$V_2^4=V_2^3\wedge V_3^4$；$U_0^3=U_2^3\vee(V_2^3\wedge U_0^2)$，$U_0^4=U_2^4\vee(V_2^4\wedge U_0^2)$；$z_0=V_0^1$，$z_1=U_0^1\oplus V_1^2$，$z_2=U_0^2\oplus V_2^3$，$z_3=U_0^3\oplus V_3^4$，$z_4=U_0^4$。总代价是20，最大深度是5。

总的来说，以习题36的表示法，代价将会是$2n+3p_n$，因为我们需要为初始的u和v得到$2n$个门，以及为前缀计算得到$3p_n$个门；由于我们不需要为$1<j\leqslant n$计算V_0^j，因此形成$z_j\,(0<j<n)$所需的$n-1$个额外的门得到了补偿。因此总代价是$14\cdot 2^m-3F_{m+5}+3$，明显优于“条件和”方法（它有同样的深度$2m+1$）：

$n=$	2	4	8	16	32	64	128	256	512	1024
“条件和”链的代价	7	25	74	197	492	1179	2746	6265	14072	31223
Ladner-Fischer链的代价	7	20	52	125	286	632	1363	2888	6040	12509

[George Boole介绍了他的代数，目的是为了证明逻辑能够按照数学来理解。最终逻辑被人们理解得如此清楚，使得情况颠倒了过来：Shannon和Zuse等人从20世纪30年代开始借助逻辑为数学设计电路，从那时开始，并行加法问题的许多方法被人们发现。第一个代价为$O(n)$、深度为$O(\log n)$的布尔链由Yu. P. Ofman设计出来（《Doklady Akademii Nauk SSSR》145 (1962), 48—51）。他的链与上述构造的链类似，但是深度大约是$4m$。]

45. 那样的论证看起来的确更简单，但是它还不足以证明所需的结果。（许多

有扇出（fan-out）0步骤的链误导人产生“这种论证更简单”的估计。）

文中的增强型排列证明技术由J. E. Savage在他的书《The Complexity of Computing》(New York: Wiley, 1976)中的定理3.4.1有介绍。

46. 当$r=2^n/n+O(1)$时，我们有$\ln(2^{2r+1}(n+r-1)^{2r}/(r-1)!)=r\ln r+(1+\ln 4)r+O(n)=(2^n/n)(n\ln 2-\ln n+1+\ln 4)+O(n)$。因此$\alpha(n)\leqslant(n/(4e))^{-2^n/n+O(n\log n)}$，当$n>4e$时它确实快速接近于0。

（事实上，(32)给定了$\alpha(11)<7.6\times 10^7$，$\alpha(12)<4.2\times 10^{-6}$，$\alpha(13)<1.2\times 10^{-38}$。）

47. 将变换限制为$(r-m)!$种情况，其中对于$1\leqslant i\leqslant n$有$i\pi=i$，$(n+r+1-k)\pi$，是第k个输出。因此，我们能用$(r-m)!c(m,n,r)\leqslant 2^{2r+1}(n+r-1)^{2r}$来代替(32)。因此，如习题46所示，当$m=O(2^n/n^2)$时几乎所有这样函数的代价都超过$2^n m/(n+\lg m)$。

48. (a)无需惊讶，当n的值较小时$C(n)$的下界相当粗糙。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r(n)$	1	1	2	3	5	9	16	29	54	99	184	343	639	1196	2246	4229

(b)“引导程序”（bootstrap）法（参见《Concrete Mathematics》§9.4）产生结果

$$r(n)=\frac{2^n}{n}\left(1+\frac{\lg n-2-1/\ln 2}{n}+O\left(\frac{\log n}{n^2}\right)\right)$$

49. 能够由长度$\leqslant r$的表达式表示的常规布尔函数的数量至多是$5^r n^{r+1} g_r$，其中g_r是有r个内部节点的有向二进制树的数量。设在这个表达式中$r=2^n/\lg n-2^{n+2}/(\lg n)^2$并除以$2^{2^n-1}$，从而在满足$L(f)\leqslant r$的这部分函数上得到一个上界。由2.3.4.4节习题7知，结果快速地趋近0，因为结果是$O((5\alpha/16)^{2^n/\lg n})$，其中$\alpha\approx 2.483$。

[J. Riordan和C. E. Shannon 为序列-并行交换网络得到了一个相似的下界（《J. Math. and Physics》21 (1942), 83—93）；这样的网络与只使用渠限化操作符的表达式等价。R. E. Krichevsky在《Problemy Kibernetiki》2(1959), 123—138中得到更一般的结果，而O. B. Lupanov在《Prob. Kibernetiki》3 (1960), 61—80中给出了一个渐近匹配的上界。]

50. (a)使用7.1.1节习题30中的子立方体表示法，那么主隐含数是00001*, (0001*1), 0100*1, 0111*1, 1010*1, 101*11, 00*011, 00*101, (01*111), 11*101, (0*1101), (1*0101), 1*1011, 0*0*11, *00101, (*01011), (*11101), 其中加上括弧的子立方体在一个最短DNF中被省略。(b)相似地，主子句和最短CNF由00111*, 01010*, 10110*, 0110**, 00*00*, 11*00*, 11*11*, (0*100*), (1*00**), 1*0*1*, (1****0), *0000*, (*1100*), *1***0, **1**0, ***1*0和(****00)给定。(因此CNF是$(x_1\vee x_2\vee\bar{x}_3\vee\bar{x}_4\vee\bar{x}_5)\wedge(x_1\vee\bar{x}_2\vee x_3\vee\bar{x}_4\vee x_5)\wedge\cdots\wedge(\bar{x}_4\vee x_6)$。)

51. $f=([x_5x_6\in\{01\}]\wedge[(x_1x_2x_3x_4)_2\in\{1,3,4,7,9,10,13,15\}])\vee([x_5x_6\in\{10,11\}]\wedge[x_1x_2x_3x_4=0000])\vee([x_5x_6\in\{11\}]\wedge[(x_1x_2x_3x_4)_2\in\{1,2,4,5,7,10,11,14\}])$。

52. 当n的值较小时的结果与那些起渐近作用的有很大不同。

n	k	l	(38)	n	k	l	(38)	n	k	l	(38)	n	k	l	(38)
5	2	2	39	8	3	2	175	11	4	4	803	14	5	5	4045

6 2 2 67	9 3 2 279	12 4 3 1329	15 5 5 7141
7 2 1 109	10 4 4 471	13 5 6 2355	16 5 4 12431

（像$[x_lx_2\in\{00,01\}]=\bar{x}_l$这样的优化经常能进一步减小代价。）

53. 首先请注意$2^k/l\leqslant n-3\lg n$，因此$m_i\leqslant n-3\lg n+1$，$2^{m_i}=O(2^n/n^3)$。同时$l=O(n)$，$t(n-k)=O(2^n/n^2)$。所以由(38)推出$l\cdot 2^{n-k}+O(2^n/n^2)=2^n/(n-3\lg n)+O(2^n/n^2)$。

54. 贪婪足迹启发法给出了一个长度为15的链：

$$\begin{array}{lll}
x_5=x_1\oplus x_2 & x_{10}=x_4\oplus x_5 & f_2=x_{15}=\bar{x}_5\wedge x_8\\
x_6=x_2\oplus x_3 & x_{11}=x_7\vee x_{10} & f_3=x_{16}=x_4\wedge \bar{x}_{12}\\
x_7=x_1\wedge \bar{x}_3 & x_{12}=x_6\oplus x_{11} & f_4=x_{17}=x_1\wedge x_8\\
x_8=x_4\wedge \bar{x}_6 & x_{13}=\bar{x}_{10}\wedge x_{12} & f_5=x_{18}=x_7\wedge x_9\\
x_9=x_4\wedge \bar{x}_5 & f_1=x_{14}=x_6\wedge \bar{x}_{11} & f_6=x_{19}=\bar{x}_3\wedge x_{13}
\end{array}$$

在我们移除掉从未使用过的步骤后，最小项优先法与一条长度为22的链相关。

$$\begin{array}{lll}
x_5=\bar{x}_1\wedge \bar{x}_2 & x_{13}=x_5\wedge x_{10} & x_{20}=x_8\wedge x_{11}\\
x_6=\bar{x}_1\wedge x_2 & x_{14}=x_5\wedge x_{11} & f_6=x_{21}=x_{15}\vee x_{18}\\
x_7=x_1\wedge \bar{x}_2 & x_{15}=x_6\wedge x_9 & f_1=x_{22}=x_{13}\vee x_{21}\\
x_8=x_1\wedge x_2 & x_{16}=x_6\wedge x_{11} & f_2=x_{23}=x_{12}\vee x_{20}\\
x_9=\bar{x}_3\wedge x_4 & x_{17}=x_7\wedge x_9 & x_{24}=x_{14}\vee x_{16}\\
x_{10}=x_3\wedge \bar{x}_4 & x_{18}=x_7\wedge x_{11} & f_3=x_{25}=x_{24}\vee x_{19}\\
x_{11}=x_3\wedge x_4 & f_5=x_{19}=x_8\wedge x_9 & f_4=x_{26}=x_{17}\vee x_{20}\\
x_{12}=x_5\wedge x_9 & &
\end{array}$$

（分配率能够以两步来替换x_{14}、x_{16}和x_{24}的计算。）

顺便说一下，习题51的答案中的三个函数能够以仅仅10步计算出来。

$$\begin{array}{lll}
x_5=x_2\vee x_4 & f_3=x_9=x_6\oplus x_8 & x_{12}=x_2\oplus x_3\\
x_6=\bar{x}_1\wedge x_5 & x_{10}=x_1\oplus x_8 & x_{13}=\bar{x}_{10}\wedge x_{12}\\
x_7=x_2\wedge x_4 & \bar{f}_2=x_{11}=x_9\vee x_{10} & f_1=x_{14}=x_4\oplus x_{13}\\
x_8=x_3\wedge \bar{x}_7 & &
\end{array}$$

55. 答案50中的最优两级DNF和CNF表示的2代价分别是53和43。当如习题54优化后，表达式(37)的代价是30。习题51中的替代方法的代价只有17。但是最优5变量链的条目清单显示对于这个6变量函数有

$$\begin{array}{llll}
x_7=\bar{x}_1\wedge x_2 & x_{11}=x_5\wedge x_{10} & x_{15}=x_{13}\oplus x_{14} & x_{18}=\bar{x}_4\wedge x_{17}\\
x_8=x_3\oplus x_7 & x_{12}=x_5\vee x_{10} & x_{16}=x_5\wedge \bar{x}_{10} & x_{19}=x_6\wedge x_{15}\\
x_9=x_2\wedge x_8 & x_{13}=x_4\wedge \bar{x}_{11} & x_{17}=\bar{x}_3\wedge x_{16} & x_{20}=x_{18}\vee x_{19}\\
x_{10}=x_1\oplus x_9 & x_{14}=x_7\wedge x_{12} & &
\end{array}$$

存在一个更好的方法吗？

56. 如果我们最多关心两个值，那么函数可以是常量、x_j或者$\bar{x}_j$。

57. 从x_5到x_{17}的真值表以十六进制表示分别是：`0ff0`，`2222`，`33cc`，`0d0d`，`7777`，`5d5d`，`3ec1`，`6b94`，`4914`，`4804`，`060b`，`2020`，`7007`。因此我们有

1010 = ，1011 = ，1012 = ，1013 = ，1014 = ，1015 = 

58. 所有代价为7的函数的真值表，如果其中刚好包含8个1，那么这个真值表

等于0779，169b或者179a。将这些以所有可能的方式组合起来会产生9656个不同结果，它们由$\{x_1,x_2,x_3,x_4\}$的排列和（或）求补运算以及$\{f_1,f_2,f_3,f_4\}$的排列和（或）求补运算产生。

59. 贪婪足迹启发法产生下列的17步链：

$$
\begin{array}{lll}
x_5 = x_1 \vee x_4 & x_{11} = x_8 \vee x_9 & x_{17} = \bar{x}_2 \wedge x_3 \\
x_6 = x_1 \oplus x_3 & x_{12} = x_1 \oplus x_{11} & f_1 = x_{18} = x_{13} \oplus x_{15} \\
x_7 = x_2 \oplus x_4 & x_{13} = x_5 \wedge \bar{x}_9 & f_2 = x_{19} = x_{11} \wedge \bar{x}_{16} \\
x_8 = \bar{x}_4 \wedge x_6 & x_{14} = x_5 \wedge x_{12} & f_3 = x_{20} = x_{12} \oplus x_{17} \\
x_9 = x_3 \oplus x_7 & x_{15} = x_2 \wedge x_6 & f_4 = x_{21} = x_{10} \wedge \bar{x}_{14} \\
x_{10} = x_2 \vee x_3 & x_{16} = x_2 \wedge \bar{x}_6 &
\end{array}
$$

初始函数都有较大的足迹，所以我们不能得到$C(f_1f_2f_3f_4)=28$；但是一个稍微更困难的S-box可能确实存在。

60. 一种方式是$u_1=x_1\oplus y_1$，$u_2=x_2\oplus y_2$，$v_1=y_2\oplus u_1$，$v_2=y_1\oplus u_2$，$z_1=v_1\wedge \overline{u}_2$，$z_2=v_2\wedge \overline{u}_1$。

61. 将这些六变量的部分函数看作4×16的真值表，其中行由x_1y_1来约束，那么我们关于4位函数的知识暗含了不错的方法来计算这些行，并产生下列的25步解法：$t_1=x_2\wedge y_2$，$t_2=x_3\wedge y_3$，$t_3=x_2\vee y_2$，$t_4=x_3\vee y_3$，$t_5=t_1\oplus t_2$，$t_6=t_1\vee t_2$，$t_7=t_4\wedge \overline{t}_5$，$t_8=t_3\oplus t_6$，$t_9=x_2\oplus y_2$，$t_{10}=t_4\oplus t_9$，$t_{11}=t_5\wedge \overline{t}_{10}$，$t_{12}=t_3\oplus t_4$，$t_{13}=x_1\vee y_1$，$t_{14}=t_8\oplus t_{12}$，$t_{15}=t_{13}\wedge \overline{t}_{14}$，$t_{16}=t_4\oplus t_7$，$t_{17}=t_{13}\wedge \overline{t}_{16}$，$t_{18}=t_3\vee t_4$，$t_{19}=x_1\oplus y_1$，$t_{20}=t_{19}\wedge \overline{t}_{18}$，$t_{21}=t_8\oplus t_{15}$，$t_{22}=t_7\oplus t_{17}$，$z_1=t_{11}\vee t_{20}$，$z_2=t_{21}\wedge \overline{t}_{20}$，$z_3=t_{22}\wedge \overline{t}_{20}$。（存在更好的方法吗？在输入是由000，001，011，101和111表示的情况下，Gilbert Lee找到了一个17步的解法。）

62. 存在$\binom{2^n}{2^n d}2^{2^n c}$个这样的函数，其中最多有$\binom{2^n}{2^n d}t(n,r)$个的代价$\leqslant r$。因此我们能够如习题46那样证明，从（32）得到结论：满足代价$\leqslant r=\lfloor 2^n c/n \rfloor$的部分最多有$2^{2r+1-2^n c}(n+r-1)^{2r}/(r-1)!=2^{-r\lg n+O(r)}$。

63. [《Problemy Kibernetiki》21(1969), 215—226。] 如Lupanov的方法那样将真值表放在一个$2^k\times 2^{n-k}$的矩阵中，并假设在列j中有c_j个关注，其中$0\leqslant j\leqslant 2^{n-k}$。将那一列分割成$\lfloor c_j/m \rfloor$个子列，使得每个都有$m$个关注，再加上一个可能为空的子列在它们底部，它包含少于m个关注。提示告诉我们最多有2^{m+k}个列向量满足每个子列的0和1的匹配，这些子列都有一个规定的首行i_0和尾行i_1。通过$O(m2^{m+3k})$个操作，我们能因此从$\{x_1,\cdots,x_k\}$的最小项构造$O(2^{m+3k})$个函数$g_t(x_1,\cdots,x_k)$，使得每个子列匹配某种类型t。并且对于每种类型t我们能够从$\{x_{k+1},\cdots,x_n\}$的最小项构造函数$h_t(x_{k+1},\cdots,x_n)$，其中规定匹配t的列；代价最多是$\sum_j(\lfloor c_j/m \rfloor+1)\leqslant 2^n c/m+2^{n-k}$。最后，$f=\bigvee_t(g_t\wedge h_t)$需要$O(2^{m+3k})$个额外的步骤。选择$k=\lfloor 2\lg n \rfloor$和$m=\lceil n-9\lg n \rceil$，使得总代价最多是$(2^n c/n)(1+9n^{-1}\lg n+O(n^{-1}))$。

当然我们需要证明提示，这个提示由E. I. Nechiporuk [《Doklady Akad. Nauk SSSR》163 (1965), 40—42]提出。事实上，$2^m(1+\lceil k\ln 2 \rceil)$个向量足够了（参见S.

K. Stein，《J. Combinatorial Theory》A16 (1974), 391—397)：如果我们随机选择 $q=2^m\lceil k\ln 2\rceil$ 个向量，而不管它们是否不同，那么未提及的子立方体的预期数量是 $\binom{k}{m}2^m(1-2^{-m})^q<\binom{k}{m}2^m e^{-q2^{-m}}<2^m$。(一个明确的构造将会更好。)

考虑一般化的情况——忽略一定百分比的错误，并规定1的比例——参见N. Pippenger，《Mathematical Systems Theory》10 (1977), 129—167。

64. 如果我们为每个格子指定一个数字，如同古代中国的魔方，那么它实际上就是tic-tac-toe游戏。[Berlekamp、Conway和Guy使用了这种计算策略来描述一个对于tic-tac-toe的完全分析，请参见他们的书《Winning Ways》3 (2003), 732—736。]

65. 一种解决方法是将“防守”步骤d_j用“进攻”步骤a_j和“反击”步骤c_j代替，并且只对角落的格子$j\in\{1,3,9,7\}$包含它们。

设$j\cdot k=(jk)\bmod 10$；那么当j表示一个角时，

$$\begin{matrix} j\cdot 1 & j\cdot 2 & j\cdot 3\\ j\cdot 4 & j\cdot 5 & j\cdot 6\\ j\cdot 7 & j\cdot 8 & j\cdot 9\end{matrix}$$

给了我们另外一种方式来看待tic-tac-toe图，因为 $j\perp 10$。那么a_j和c_j的精确定义是

$$\begin{aligned} a_j &= m_j\wedge\bigl((x_{j\cdot 3}\wedge\beta_{(j\cdot 8)(j\cdot 9)}\wedge(o_{j\cdot 4}\oplus o_{j\cdot 6}))\vee(x_{j\cdot 7}\wedge\beta_{(j\cdot 6)(j\cdot 9)}\wedge(o_{j\cdot 2}\oplus o_{j\cdot 8}))\\ &\qquad\vee(m_{j\cdot 9}\wedge((m_{j\cdot 8}\wedge x_{j\cdot 2}\wedge\overline{(o_{j\cdot 3}\oplus o_{j\cdot 6})})\vee(m_{j\cdot 6}\wedge x_{j\cdot 4}\wedge\overline{(o_{j\cdot 7}\oplus o_{j\cdot 8})})))\bigr);\\ c_j &= d_j\wedge\overline{(x_{j\cdot 6}\wedge o_{j\cdot 7})}\wedge\overline{(x_{j\cdot 8}\wedge o_{j\cdot 3})}\wedge\bar d_{j\cdot 9};\end{aligned}$$

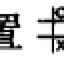
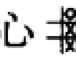
这里 $d_j=m_j\wedge\beta_{(j\cdot 2)(j\cdot 3)}\wedge\beta_{(j\cdot 4)(j\cdot 7)}$ 代替(51)。我们也定义

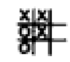
$$\begin{aligned} u&=(x_1\oplus x_3)\oplus(x_7\oplus x_9),\\ v&=(o_1\oplus o_3)\oplus(o_7\oplus o_9),\\ t&=m_2\wedge m_6\wedge m_8\wedge m_4\wedge(u\vee\bar v),\end{aligned}\qquad z_j=\begin{cases} m_j\wedge\bar t, & \text{如果 } j=5\\ m_j\wedge\bar d_{j\cdot 9}, & \text{如果 } j\in\{1,3,9,7\}\\ m_j, & \text{如果 } j\in\{2,6,8,4\}\end{cases}$$

以便涉及一些更多的例外情况。最终在（53）中按等级排列的移动序列$d_5d_1d_3d_9d_7d_2d_6d_8d_4m_5m_1m_3m_9m_7m_2m_6m_8m_4$由序列$a_1a_3a_9a_7c_1c_3c_9c_7z_5z_1z_3z_9z_7z_2z_6z_8z_4$替代；当$j$为角落格子时，我们将(55)中的 $(d_j\wedge\bar d_j')\vee(m_j\wedge\bar m_j')$ 用 $(a_j\wedge\bar a_j')\vee(c_j\wedge\bar c_j')\vee(z_j\wedge\bar z_j')$ 替代，否则简单地由 $(z_j\wedge\bar z_j')$ 替代。

(请注意这台机器要求能从所有合法的位置正确地移动，甚至当机器执行完×更早的移动后，那些位置却不能出现时也一样。我们实质上允许人们来玩这个游戏，直到他们向机器寻求建议。否则大幅度的简化也是有可能的。例如，如果×一直先行，那么它能够夺得中间的格子，并消除大量将来的可能情况；能够产生少于8×6×4×2=384盘的游戏。即使是○先行，那么也只有少于9×7×5×3=945种可能的情形违背一个固定的策略。事实上，使用这里定义的策略的不同游戏场次的实际数量是76+457，其中72+328次由机器赢得，其余的属于平局。)

66. 在前面答案中的布尔链完成了它的任务，即从所有4520个合法的位置做正

确的移动，其中正确性实际的意思是最坏情况的最终结果被最大化。但是一个真正伟大的tic-tac-toe玩家会以不同的方式来处理。例如，机器从位置得到中心，并且○可能会在角落里进行下一步，从而打成平局。但是移动到仅仅能给○两个机会来避免失败。[参见Martin Gardner，《Hexaflexagons and Other Mathematical Diversions》，Chapter 4。]

进一步来说，从一个位置如的最佳移动是到，而不是立即获得胜利；然后如果应答是，那么移动到你仍然赢了，只不过没有让你的对手那么丢脸罢了。

最后，甚至单单一步"最佳移动"的概念也是有问题的，因为一个好的玩家会在不同的游戏场次里选择不同的移动方法（正如Babbage观察到的）。

> 人们可能会认为在一个数字计算机上编程来玩ticktacktoe，或者为一台ticktacktoe机器设计专门的电路，会是一件容易的事情。除非你的目的是构造一个大师级的机器人，它能在与无经验的玩家的比赛中赢得最多数量的场次，那么这种观点才是正确的。
>
> ——Martin Gardner，《The Scientific American Book of Mathematical Puzzles & Diversions》(1959)

67. 作者最好的一次尝试是采用了答案63中的Sholomov方法来构造，其中使用了1734个门：首先将真值表对$o_5x_5o_2o_6o_8o_4$划分成64行，对于其他12个输入变量划分成4096列。然后将合适的1放到"关注"的位置，通过这种方式使得这些列中1的数量相对很少。然后找到少量的列类型，使得它们与所有列中的"关注"匹配；23种类型满足c函数，20种类型满足s，6种满足m。然后我们可以计算每个输出为$\vee(g_i \wedge h_i)$，通过g_i和h_i完成最小项计算工作中的一大部分。

[这道习题是受到John Wakerly的一本书《Digital Design》(Prentice-Hall, 3rd edition, 2000), 6.2.7节中的一个讨论的启发而构造的。顺便说一句，Babbage打算通过考察$N \bmod k$的值来从k种可能的移动中选择，其中N是目前已赢得的局数；他没有认识到连续的移动之间会是高度相关的，直到N的值改变为止。更好的办法是让N代表目前执行过的移动数。]

68. 不是。这种方法生成了一条"统一"的链，它具有易于理解的结构，但是它的代价属于n的2^n次多项式级别。存在一个由定理L构造的拥有约$2^n/n$个门的电路，但是这样一个电路更难构造。（顺便说一句，$C(\pi_5)=10$。）

69. (a)例如，可以通过尝试所有64种情况来验证这个结果。

(b)如果x_m与x_i位于同一行或列，且与x_j位于同一行或列，那么我们有$\alpha_{111}=\alpha_{101}=\alpha_{110}=0$，因此这些函数对是"好"的。否则，实质上存在3种不同的可能情况，都是"坏"的：如果$(i,j,m)=(1,2,4)$，那么$\alpha_{101}=0$，$\alpha_{100}=x_5x_9\oplus x_6x_8$，$\alpha_{011}=x_9$；如果$(i,j,m)=(1,2,6)$，那么$\alpha_{010}=x_4x_9$，$\alpha_{011}=x_7$，$\alpha_{100}=x_5x_9$，$\alpha_{101}=x_8$；如果$(i,j,m)=(1,5,9)$，那么$\alpha_{111}=1$，$\alpha_{110}=0$，$\alpha_{010}=x_3x_7$。

70. (a) $x_1 \wedge ((x_5 \wedge x_9) \oplus (x_6 \wedge x_8) \oplus x_2 \wedge ((x_6 \wedge x_7) \oplus (x_4 \wedge x_9)) \oplus x_3 \wedge ((x_4 \wedge x_8) \oplus (x_5 \wedge x_7))$。

(b) $x_1 \wedge ((x_5 \wedge x_9) \vee (x_6 \wedge x_8) \vee x_2 \wedge ((x_6 \wedge x_7) \vee (x_4 \wedge x_9)) \vee x_3 \wedge ((x_4 \wedge x_8) \vee (x_5 \wedge x_7))$。

(c)设 $y_1 = x_1 \wedge x_5 \wedge x_9$， $y_2 = x_1 \wedge x_6 \wedge x_8$， $y_3 = x_2 \wedge x_6 \wedge x_7$， $y_4 = x_2 \wedge x_4 \wedge x_9$， $y_5 = x_3 \wedge x_4 \wedge x_8$， $y_6 = x_3 \wedge x_5 \wedge x_7$。函数$f(y_1,\cdots,y_6)=[y_1+y_2+y_3>y_4+y_5+y_6]$能够使用两个完全加法器和一个比较器通过继续执行15步计算出来；但是存在一个14步的解法：设$z_1=(y_1\oplus y_2)\oplus y_3$，$z_2=(y_1\oplus y_2)\vee(y_1\oplus y_3)$，$z_3=(y_4\oplus y_5)\oplus y_6$，$z_4=(y_4\oplus y_5)\vee(y_4\oplus y_6)$。那么$f=(z_1\oplus(z_2\wedge(\bar z_4\oplus(z_1\vee z_3))))\wedge(\bar z_3\vee z_4)$。进一步，$y_1y_2y_3=111\Leftrightarrow y_4y_5y_6=111$；因此存在“不关注”，从而产生一个11步的解法：$f=((\bar z_1\wedge z_3)\vee \bar z_4)\wedge z_2$。总代价是12+11=23。

（在仅给定f的真值表的情况下，作者不知道任何一个能让计算机在合理的时间内发现这样一条有效链的方法。但是有可能存在一条甚至更好的链。）

71. (a)$P(p)=1-12p^2+24p^3+12p^4-96p^5+144p^6-96p^7+24p^8$，当 $p=\frac{1}{2}+\varepsilon$时，它的值是 $\frac{11}{32}+\frac{9}{2}\varepsilon^2-3\varepsilon^4-24\varepsilon^6+24\varepsilon^8$。

(b)存在8个值$(f_0,\cdots,f_7)$的$N=2^{n-3}$种值设定，每一种生成“好”的函数对的概率是$P(p)$。因此答案是$1-P(p)^N$。

(c)概率是 $\binom{N}{r}P(p)^r(1-P(p))^{N-r}$，其中恰有$r$种设定是成功的；而且在这样的情况下$t$次尝试找到“好”的函数对的概率是$(r/N)^t$。答案因此是 $1-\sum_{r=0}^{N}\binom{N}{r}P(p)^r(1-P(p))^{N-r}(r/N)^t=1-P(p)^t+O(t^2/N)$。

(d) $\sum_{r=0}^{N}\binom{N}{r}P(p)^r(1-P(p))^{N-r}\sum_{j=0}^{t-1}(r/N)^j=(1-P(p)^t)/(1-P(p))+O(t^3/N)$。

72. 习题71(a)中的可能的概率为$P(p)+(72p^3-264p^4+432p^5-336p^6+96p^7)r+(60p^2-240p^3+456p^4-432p^5+144p^6)r^2+(-48p^2+144p^3-216p^4+96p^5)r^3+(-36p^2+24p^3+12p^4)r^4+(48p^2-24p^3)r^5-12p^2r^6$。如果$p=q=(1-r)/2$，那么它是$(11+48r+36r^2-144r^3-30r^4+336r^5-348r^6+144r^7-12r^8)/32$；例如，当$r=1/2$时，它是$7739/8192\approx0.94$。

73. 考虑Horn子句$1\wedge2\Rightarrow3$，$1\wedge3\Rightarrow4$，…，$1\wedge(n-1)\Rightarrow n$，$1\wedge n\Rightarrow2$和$i\wedge j\Rightarrow1$，其中$1<i<j\leqslant n$。假设在一次分解中有$|Z|>1$，且设$i$为满足$x_i\in Z$的最小值。同时设$j$为满足$j>i$和$x_j\in Z$的最小值。我们无法得到$i>1$，因为在那种情况下有$i\wedge j\Rightarrow1$。因此$i=1$，$x_j\in Z$，其中$2\leqslant j\leqslant n$。

74. 假设我们知道没有非平凡分解能与$x_i\in Z$或……或$x_{i-1}\in Z$共存；最初$i=1$。我们也希望通过聪明地选择j和m来排除$x_i\in Z$。当i被断言时，Horn子句$i\wedge j\Rightarrow m$归纳为Krom子句$j\Rightarrow m$。因此我们实质上想要在一个有向图中为强分图使用Tarjan的深度优先搜索，在图中弧$j\Rightarrow m$可能存在，也可能不存在。

当从点j开始探索时，首先尝试$m=1,\cdots,m=i-1$；如果任何这样的推断$i\wedge j\Rightarrow m$能成立，那么我们可以将j和它的所有前趋从对于i的有向图中除去。否则，对于任何

这样被除去的顶点m，测试$j\Rightarrow m$是否成立。否则测试未探索的顶点m。否则尝试那些已经见过的顶点m，优先考虑那些靠近深度优先树的根的点。

在例子$f(x)=(\det X) \bmod 2$中，我们会连续地得到$1\wedge 2\not\Rightarrow 3$，$1\wedge 2\Rightarrow 4$，$1\wedge 4\Rightarrow 3$，$1\wedge 3\Rightarrow 5$，$1\wedge 5\Rightarrow 6$，$1\wedge 6\Rightarrow 7$，$1\wedge 7\Rightarrow 8$，$1\wedge 8\Rightarrow 9$，$1\wedge 9\Rightarrow 2$（现在$i\leftarrow 2$）；$2\wedge 3\not\Rightarrow 1$，$2\wedge 3\Rightarrow 4$，$2\wedge 4\not\Rightarrow 1$，$2\wedge 4\not\Rightarrow 5$，$2\wedge 4\Rightarrow 6$，$2\wedge 6\Rightarrow 1$（现在3、4和6从对于2的有向图中被除去），$2\wedge 5\Rightarrow 1$（5被除去），$2\wedge 7\not\Rightarrow 1$，$2\wedge 7\Rightarrow 3$（7被除去），$2\wedge 8\Rightarrow 1$，$2\wedge 9\Rightarrow 1$（现在$i\leftarrow 3$）；$3\wedge 4\not\Rightarrow 1$，$3\wedge 4\Rightarrow 2$，$3\wedge 5\Rightarrow 1$等。

75. 这个函数仅在两点处的值等于1，它们是互补的。因此它是不能分解的；然而当$n>3$时函数对(58)不可能是“坏的”。因此每个划分(Y,Z)都是分解的一个候选。

相似地，如果f关于(Y,Z)可分解，那么不可分解的函数$f(x)\oplus S_{0,n}(x)$与f在测试中会有相似的效果。（一个处理“近似可分解函数”的方法或许应该在一个一般目的的可分解性测试器中提供。）

76. (a)设$a_l=[i\geqslant l]$，其中$0\leqslant l\leqslant 2^m$。代价$\leqslant 2t(m)$，正如在答案38（$b$）中观察到的那样；而且事实上，代价可以以$\Theta(m)$的深度减少到$2^{m+1}-2m-2$。进一步，函数$[i\leqslant j]=(\bar{i}_1\wedge j_1)\vee((i_1\equiv j_1))\wedge[i_2\cdots i_m\leqslant j_2\cdots j_m])$能够以$4m-3$个门计算出来。在计算出$x\oplus y$后，每个$z_l$的代价是$2^{m+1}+1=O(n/\log n)$。

(b)在这里，由定理L知代价最多是$C(g_0)+\cdots+C(g_{2^m})\leqslant(2^m+1)(2^{2^m}/(2^m+O(m)))$，因为每个$g_l$是一个有$2^m$个输入的函数。

(c)如果$i\leqslant j$，那么对于$l\leqslant i$有$z_l=x$，对于$l>i$有$z_l=y$；因此$f_i(x)=c_0\oplus\cdots\oplus c_i$，$f_j(y)=c_{j+1}\oplus\cdots\oplus c_{2^m}$。如果$i>j$，那么对于$l\leqslant i$有$z_l=y$，对于$l>i$有$z_l=x$；因此$f_j(y)=c_0\oplus\cdots\oplus c_j$，$f_i(x)=c_{i+1}\oplus\cdots\oplus c_{2^m}$。

(d)函数$b_l=[j<l]$可以以$O(2^m)$步计算出来，其中$0\leqslant l\leqslant 2^m$，如同(a)中一样。因此我们可以由$(c_0,\cdots,c_{2^m})$通过$O(2^m)$个更多的门计算出$F$来。因此步骤(b)对于$m$的值较大的情况在代价上有优势。

(e) $a_0=1$，$a_1=i$，$a_2=0$；$b_0=0$，$b_1=j$，$b_2=1$；$d=[i\leqslant j]=\bar{i}\vee j$；$m_l=a_l\oplus d$，$z_{l0}=x_0\oplus(m_l\wedge(x_0\oplus y_0))$，$z_{l1}=x_1\oplus(m_l\wedge(x_1\oplus y_1))$，其中$l=0,1,2$；$c_0=z_{01}$；$c_1=z_{10}\wedge\bar{z}_{11}$；$c_2=z_{10}\vee z_{21}$；$c'_l=c_l\wedge(d\equiv a_l)$，$c''_l=c_l\wedge(d\equiv b_l)$，其中$l=0,1,2$；最后$F=(c'_0\oplus c'_1\oplus c'_2)\vee(c''_0\oplus c''_1\oplus c''_2)$。

当然，净代价（在明显的简化后是29）在这样一个小例子中显得令人无法容忍。但是人们会好奇是否有流行的自动优化程序能够将这条链精简到只有5个门。

[这个结果是《Matematicheskie Zametki》15(1974)，937—944；《London Math. Soc. Lecture Note Series》169 (1992)，165—173中一些更一般定理的一种特殊情况。]

77. 对于f_n或$\bar{f}_n$，给定一条这样的最短的链，设$U_l=\{i\mid l=j(i)或l=k(i)\}$为对$x_l$的“使用”，并且设$u_l=|U_l|$。如果$x_i=x_{j(i)}\vee x_{k(i)}$，那么设$t_i=1$；否则$t_i=0$。我们将证明存在一条长度$\leqslant r-4$的链，它能计算$f_{n-1}$或者$\bar{f}_{n-1}$，方法是利用如下的思想：如果对于任意$m$，变量$x_m$的值设为0或1，那么我们可以为$f_{n-1}$或$\bar{f}_{n-1}$得到一条链，方法是通过删除$U_m$的所有步骤，并且合理地修改其他步骤。进一步说，当$x_m$的值已经设为0或1

时，如果$x_i=x_{j(i)}\circ x_{k(i)}$且$x_{j(i)}$或$x_{k(i)}$中有一个与$t_i$相等，那么我们也将步骤$U_i$删除掉。（在这个论证的过程中，字母$m$将代表位于范围$1\leqslant m\leqslant n$内的一个索引。）

情况1：对于一些m，$u_m=1$。这种情况不可能发生在一条最短链里面。因为如果对x_m唯一的使用是$x_i=\bar{x}_m$，那么除去这一步会改变$f_n\leftrightarrow\bar{f}_n$；否则我们可以设定$x_1,\cdots,x_{m-1},x_{m+1},\cdots,x_n$的值，使得$x_i$与$x_m$独立，与$x_{n+r}=f_n$或$\bar{f}_n$相矛盾。因此每个变量必须使用至少两次。

情况2：对于一些l和m，$x_i=\bar{x}_m$，其中$u_m>1$。那么对于一些i和k有$x_i=x_l\circ x_k$，而且我们能够设$x_m\leftarrow\bar{t}_i$，使得x_i独立于x_k。除去步骤U_m、U_l和U_i能够减少至少4步，除非当$u_l=u_i=1$且$u_m=2$且$x_j=x_m\circ x_i$时；但是那种情况下我们也可以除去U_j。

情况3：对于一些m，$u_m\geqslant 3$，同时不属于情况2。如果$i,j,k\in U_m$且$i<j<k$，那么设$x_m\leftarrow t_k$并删除步骤i，j，k，U_k。

情况4：$u_1=u_2=\cdots=u_n=2$，同时不属于情况2。我们可以假定第一步是$x_1=x_1\circ x_2$，对于一些$k<l$有$x_l=x_1\circ x_k$。

情况4.1：$k>0$。那么$k>1$。如果$u_k=1$，那么设$x_1\leftarrow t_l$并移除步骤1，k，l，U_l。否则设$x_2\leftarrow t_1$；从而可以推出$x_k=\bar{t}_l$，并且我们可以移除步骤1，k，l，U_k。

情况4.2：$x_l=x_1\circ x_m$。那么我们肯定有$m=2$；因为如果$m>2$，那么我们可以设$x_2\leftarrow t_1$，$x_m\leftarrow t_l$，并让x_r与x_1独立。因此我们可以假定$x_1=x_1\wedge x_2$，$x_2=x_1\vee x_2$。通过设$x_1\leftarrow 0$，我们就可以移除U_0和U_1；通过设$x_1\leftarrow 1$，我们就可以移除U_0和U_2。因此我们完成了任务，除非$u_1=u_2=1$。

如果$x_p=\bar{x}_1$，那么设$x_1\leftarrow 0$并移除1，2，p，U_p；如果$x_q=\bar{x}_2$，设$x_1\leftarrow 1$并移除1，2，q，U_q。否则$x_p=x_1\circ x_u$且$x_q=x_2\circ x_v$，其中x_u和x_v不依赖x_1或x_2。但是那是不可能的事情；我们可以通过设$x_3,\cdots,x_n$来得到$x_u=t_p$，然后设$x_2\leftarrow 1$来让x_r独立于x_1。

[《Problemy Kibernetiki》**23** (1970), 83—101；**28** (1974)，4。通过相似的证明方法，Red'kin证明了对函数“$x_1\cdots x_n<y_1\cdots y_n$”和“$x_1\cdots x_n=y_1\cdots y_n$”的最短“且-或-非”链的长度分别是$5n-3$和$5n-1$。]

78. [《SICOMP》**6**(1977), 427—430。]如果$k\in S$，那么称y_k是活跃的。我们可以假定这条链是正规的，且$\|S\|>1$；证明与答案77中Red'kin所做的类似。

情况1：一些活跃的y_k被用到不止一次。设$y_k\leftarrow 0$能够至少节省两步，并为一个有$\|S\|-1$个活跃的值的函数生成一条链。

情况2：一些活跃的y_k只在一个AND门中出现。设$y_k\leftarrow 0$能够至少削减两步，除非这个AND是最后一步。但是它不可能是最后一步，因为$y_k=0$使得结果独立于每个其他的活跃的y_j。

情况3：与情况2类似，但是有一个OR门或NOT-BUT门或BUT-NOT门。对于一些合适的常数c，通过设$y_k\leftarrow c$能达到所需的效果。

情况4：与情况2类似，但是有XOR门。这个门不可能是最后的，因为当$(x_1\cdots x_m)_2$代表一个不同的活跃的值y_j的地址时，结果应该与y_k独立。因此通过将y_k设为由另一个XOR的输入定义的函数，我们可以削减两步。

79. (a)假设代价是$r<2n-2$；然后$n>1$。如果每个变量只使用一次，那么两个叶

子节点必须是兄弟节点。因此一些变量需要用到至少两次。将它删除会生成一条$n-1$个变量的代价$\leqslant r-2$的链，它上面不存在兄弟节点。

（顺便说一句，如果每个变量都使用至少两次，那么代价至少是$2n-1$，因为至少$2n$次变量的使用必须在链中连接起来。）

(b) 请注意当边u—v在$\{x_1,\cdots,x_n\}$上构成了一棵自由的树时，都有 $S_{0,n}=\bigwedge_{u—v}(u\equiv v)$，因此存在许多方式能达到$2n-3$的代价。

任何代价为$r<2n-3$的链肯定有$n>2$，且肯定包含兄弟节点u和v。通过重命名和可能补充中间结果，我们可以假定$u=1$，$v=2$，$f(x_1,\cdots,x_n)=g(x_1\circ h(x_3,\cdots,x_n),x_2,\cdots,x_n)$，其中$\circ$是$\wedge$或$\oplus$。

情况1：$\circ$是AND。我们肯定有$h(0,\cdots,0)=h(1,\cdots,1)=1$，因为否则$f(x_1,x_2,y,\cdots,y)$就不会依赖$x_1$了。因此$f(x_1,\cdots,x_n)=h(x_3,\cdots,x_n)\wedge g(x_1,x_2,\cdots,x_n)$可以通过一条同样代价的链计算出来，在这条链中1和2是兄弟节点，且它们之间的路径更短一些。

情况2：$\circ$是XOR。那么$f=f_0\vee f_1$，其中$f_0(x_1,\cdots,x_n)=(x_1\equiv h(x_3,\cdots,x_n))\wedge g(0,x_2,\cdots,x_n)$且$f_1(x_1,\cdots,x_n)=(x_1\oplus h(x_3,\cdots,x_n))\wedge g(0,x_2,\cdots,x_n)$。但是$f=S_{0,n}$只有两个为素数的蕴涵项；因此仅存在四种可能性。

情况2a：$f_0=f$。那么我们可以用0来代替$x_1\oplus h$，为函数$g(0,x_2,\cdots,x_n)=S_{0,n-1}(x_2,\cdots,x_n)$得到一条代价$\leqslant r-2$的链。

情况2b：$f_1=f$。与情况2a相似。

情况2c：$f_0(x)=x_1\wedge\cdots\wedge x_n$且$f_1(x)=\bar{x}_1\wedge\cdots\wedge\bar{x}_n$。在这种情况下我们肯定有$g(0,x_2,\cdots,x_n)=x_2\wedge\cdots\wedge x_n$和$g(1,x_2,\cdots,x_n)=\bar{x}_2\wedge\cdots\wedge\bar{x}_n$。因此将$h$用1替换会产生一条链，它可以以小于$r$步计算出$f$。

情况2d：$f_0(x)=\bar{x}_1\wedge\cdots\wedge\bar{x}_n$且$f_1(x)=x_1\wedge\cdots\wedge x_n$。与情况2c类似。

反复应用这些归纳会导致矛盾。相似地，人们可以证明$C(S_0S_n)=2n-2$。[《Theoretical Computer Science》1 (1976), 289—295。]

80. (a)[《Mathematical Systems Theory》10 (1977), 323—336。]不失一般性，$a_0=0$且链是正规的。如答案77中一样定义U_l和u_l。我们可以由对称性假定$u_1=\max(u_1,\cdots,u_n)$。

我们肯定有$u_l\geqslant 2$。因为如果$u_1=1$，我们可以进一步假定$x_{n+1}=x_1\circ x_2$；因此三个函数$S_\alpha(0,0,x_3,\cdots,x_n)=S_{\alpha''}$，$S_\alpha(0,1,x_3,\cdots,x_n)=S_{'\alpha'}$，$S_\alpha(1,1,x_3,\cdots,x_n)=S_{''\alpha}$中会有两个相等。但是那样$S_\alpha$会成为一个奇偶性函数，或者$S_{'\alpha'}$会是常量。

因此通过设$x_1=0$，我们可以除去U_1的门，为S_α得到一条门的数量至少要少于2的链。这就意味着$C(S_\alpha)\geqslant C(S_{\alpha'})+2$。相似地，设$x_1=1$，可以证明$C(S_\alpha)\geqslant C(S_{'\alpha})+2$。

当我们进一步探索这个问题时，会出现3种情况。

情况1：$u_1\geqslant 3$。设$x_1=0$，可以证明$C(S_\alpha)\geqslant C(S_{\alpha'})+3$。

情况2：$U_1=\{i,j\}$且操作符$\circ_j$是渠限化的（亦即AND、BUT-NOT、NOT-BUT或者OR）。通过将x_1设为一个合适的常量，强置x_j的值，并让我们可以除去$U_1\cup U_j$；请注意在一条最优链里面有$i\notin U_j$。因此要么有$C(S_\alpha)\geqslant C(S_{\alpha'})+3$，要么有$C(S_\alpha)\geqslant C(S_{'\alpha})+3$。

情况3：$U_1=\{i,j\}$且$o_i=o_j=\oplus$。我们可以假定$x_i=x_1\oplus x_2$且$x_j=x_1\oplus x_k$。如果$u_j=1$且$x_l=x_j\oplus x_p$，那么我们可以通过设$x_j=x_k\oplus x_p$，$x_l=x_1\oplus x_j$来重构这条链；因此我们可以假定对于一些渠限化操作符∘要么有$u_j\neq 1$，要么有$x_l=x_j\circ x_p$。如果$U_2=\{i,j'\}$，那么我们可以相似地假定$x_{j'}=x_1\oplus x_{k'}$，且对于一些渠限化操作符∘'要么有$u_{j'}=1$，要么有$x_{l'}=x_{j'}\circ' x_{p'}$。进一步，我们可以由对称性假定$x_j$不依赖于$x_{j'}$。

如果x_k不依赖x_i，那么设$f(x_3,\cdots,x_n)=x_k$；否则当$x_i=1$时设$f(x_3,\cdots,x_n)$为x_k的值。通过设$x_1=f(x_3,\cdots,x_n)$和$x_2=\overline{f}\,(x_3,\cdots,x_n)$，或者反过来，我们使得$x_i$和$x_j$为常量，而且我们为非常量函数$S_{\cdot\alpha}$得到了一条链。事实上我们可以保证在情况$u_j=1$中$x_l$为常量。我们断言这条链至少有5个门（包括$x_i$和$x_j$）可以被除去；因此$C(S_\alpha)\geqslant C(S_{\cdot\alpha})+5$。如果$|U_i\cup U_j|\geqslant 3$，那么这个断言明显是正确的。

我们肯定有$|U_i\cup U_j|>1$。否则我们有$p=i$，而且x_k不依赖x_i，因此S_α将因为我们对x_2的选择而独立于x_1。因此$|U_i\cup U_j|=2$。

情况3a：$U_j=\{l\}$。那么x_l是常量；我们可以删掉x_i、x_j以及$U_i\cup U_j\cup U_l$。如果后一个集合仅包含两个元素，那么$x_q=x_i\circ x_l$也是常量，于是我们将U_q删掉。因为$S_{\cdot\alpha}$不是常量，所以我们将不会删除输出的门。

情况3b：$U_i\subsetneq U_j$，$|U_j|=2$。那么对于一些q有$x_q=x_i\circ x_j$；我们可以除去x_i、x_j以及$U_j\cup U_q$。这个断言已经得到证明。

(b) 通过归纳，$C(S_k)\geqslant 2n+\min(k,n-k)-3-[n=2k]$，其中$0<k<n$；$C(S_{\geqslant k})\geqslant 2n+\min(k,n+1-k)-4$，其中$1<k<n$。简单的情况是$C(S_0)=C(S_n)=C(S_{\geqslant 1})=C(S_{\geqslant n})=n-1$；$C(S_{\geqslant 0})=0$。(根据图9和图10，当$n\leqslant 5$时，对于$k=\lceil n/2\rceil$这些边界是最优的。所有已知的结果与推测$C(S_k)=C(S_{\geqslant k})$是一致的，其中$k\geqslant n/2$。)

81. 如果一些变量使用的次数多于1次，那么我们可以将它设为常量，使得n减少1，且c减少$\geqslant 2$。否则第一个操作必须包含x_1，因为$y_1=x_1$是唯一不需要计算的输出；通过让x_1成为常量，使得n减少1，c减少$\geqslant 1$，d减少$\geqslant 1$。[《J. Algorithms》7 (1986), 185—201。]

82. (62)是错误的。

(63)读作“对于所有的数字m，存在一个数n使得$m<n+1$”；它是正确的，因为我们可以取$m=n$。

(64)当$n=0$或者$n=1$时不成立，因为这些式子中的数字需要是非负整数。

(65)就是说，如果b比a大2或更多，那么存在一个数字ab在它们之间。当然它是正确的，因为我们可以取$ab=a+1$。

(66)在正文中有解释，它也是正确的。请注意“∧”的优先级高于“∨”，且“≡”的优先级高于“⇔”，正如在(65)中“+”的优先级高于“⩾”，且“<”的优先级高于“∧”；这些规律减少了在L的语句里使用括号的需要。

(67)就是说，如果A包含至少一个元素n，那么它必须包含一个最小元素m（一个小于或等于它的所有元素的元素）。正确。

(68)情况相似，但是m现在是一个最大元素。这种情况也是正确的，因为所有的集合都假定是有限的。

(69)要求一个集合P，它具有的性质是$[0\in P]=[3\notin P]$，$[1\in P]=[4\notin P]$，…，$[999\in P]=[1002\notin P]$，$[1000\in P]\neq[1003\notin P]$，$[1001\in P]\neq[1004\notin P]$等。它在当（且仅当）$P=\{x|x \bmod 6\in\{1,2,3\}$且$0\leqslant x<1000\}$时正确。

最后，(70)中的子表达式$\forall n(n\in C\Leftrightarrow n+1\in C)$是用另一种方式说明$C=\emptyset$，因为$C$是有限的。因此在$\forall A\forall B$后的带括号的式子是用一种俏皮的方式说明$A=\emptyset$且$B\neq\emptyset$。(Stockmeyer和Meyer不止一次地使用这种技巧缩写了L中涉及长子表达式的语句。)(70) 是正确的，因为一个空集不等于一个非空集合。

83. 我们可以假定这条链是正规的。设渠限化步骤是$y_1,\cdots,y_p$。那么$y_k=\alpha_k\circ\beta_k$且$f=\alpha_{p+1}$，其中α_k和β_k是$\{x_1,\cdots,x_n,y_1,\cdots,y_{k-1}\}$的一些子集的$\oplus$运算；要计算它们，最多需要$n+k-2$个$\oplus$，首先结合共同的项。因此$C(f)\leqslant p+\sum_{k=1}^{p+1}(n+k-2)=(p+1)(n+p/2)-1$。

84. 与前面的答案中的证明相似，用$\vee$或$\wedge$代替$\oplus$。[N. Alon和R. B. Boppana,《Combinatorica》7 (1987), 15—16。]

85. (a)一个简单的计算机程序显示13744是合法的，19024是非法的。(一个这种非法的组至少有8个成员；例如{00,0f,33,55,ff,15,3f,77}。事实上，如果函数$x_1\vee x_2$(3f)，$x_2\vee x_3$(77)和$(x_1\vee x_2)\wedge x_3$(15)在一个合法的组L中存在，那么$x_2\sqcup$15=33|15=37必须也在L里面。)

(b)投影函数和常量函数明显存在。定义$A^*=\cap\{B|B\supseteq A$且$B\in\mathcal{A}\}$，或者$A^*=\infty$，如果没有这样的集合B存在的话。那么我有$\lceil A\rceil\sqcap\lceil B\rceil=\lceil A\cap B\rceil$且$\lceil A\rceil\sqcup\lceil B\rceil=|(A\cup B)^*|$。

(c)简化表达式为$\hat{x}_l\subseteq x_l\vee\bigvee_{i=n+1}^{l}$，$x_l\subseteq\hat{x}_l\vee\bigvee_{i=n+1}^{l}\varepsilon_i$，并由归纳证明：如果步骤$l$是一个AND步骤，那么$\hat{x}_l=\hat{x}_j\sqcap\hat{x}_k\subseteq\hat{x}_j\wedge\hat{x}_k\subseteq\left(x_j\vee\bigvee_{i=n+1}^{l}\delta_i\right)\wedge\left(x_k\vee\bigvee_{i=n+1}^{l}\delta_i\right)=x_l\vee\bigvee_{i=n+1}^{l}\delta_i$；$x_l=x_j\wedge x_k\subseteq\left(\hat{x}_j\vee\bigvee_{i=n+1}^{l-1}\varepsilon_i\right)\wedge\left(\hat{x}_k\vee\bigvee_{i=n+1}^{l-1}\varepsilon_i\right)=(\hat{x}_j\wedge\hat{x}_k)\vee\bigvee_{i=n+1}^{l-1}\varepsilon_i$，且$\hat{x}_j\wedge\hat{x}_k=\hat{x}_l\vee\varepsilon_l$。如果步骤$l$是一个OR步骤，那么以相似的方式证明。

86. (a)如果S是一个包含在$(r+1)$组S'中的r组，那么明显有$\Delta(S)\subseteq\Delta(S')$。

(b)由鸽笼原理知，当S为r组时，$\Delta(S)$包含每一部分的元素u和v。而且如果$\Delta(S)=\{u,v\}$，那么我们肯定有u—v。

(c)当$r=1$时结果很明显。由“强”性质知，存在至多$r-1$条边包含任意给定的顶点u。如果u—v，那么与$\{u, v\}$不相交的边是强$(r-1)$闭的；所以由归纳知最多存在它们中的$(r-2)^2$个。因此总共最多存在$1+2(r-2)+(r-2)^2$条边。

(d)是的，由习题85(b)知，如果$r>1$，因为强r闭的图在交集下是闭合的。当$r>1$时，所有拥有的边数$\leqslant 1$的图都是强r闭的，因为它们没有r组能包含不同的边。

(e)存在$\binom{n}{3}$个三角形$x_{ij}\wedge x_{ik}\wedge x_{jk}$，它们中只有$n-2$个是包含在$\hat{f}$的任何项$x_{uv}$里。因此对于最多$(r-1)^2(n-2)$个三角形的最小项被包含在$\hat{f}$里，而且其他项肯定被包含

在函数 $\varepsilon_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \wedge x_{k(i)})$ 中的一个里。这样的一项的形式为 $T = (\lceil G \rceil \sqcap \lceil H \rceil) \oplus (\lceil G \rceil \wedge \lceil H \rceil) = (\lceil G \rceil \wedge \lceil H \rceil) \wedge \overline{\lceil G \cap H \rceil}$，其中$G$和$H$是强$r$闭的；我们将要证明$T$包含最多$2(r-1)^3$个三角形。

一个T中的三角形$x_{ij} \wedge x_{ik} \wedge x_{jk}$必须涉及$\lceil G \rceil$的一些变量($x_{ij}$)和$\lceil H \rceil$的一些变量($x_{ik}$)，但是不能涉及$\lceil G \cap H \rceil$的变量。对于$ij$存在着最多$(r-1)^2$种选择，那么对于$k$存在着最多$2(r-1)$种选择，因为$H$最多有$r-1$条边涉及$i$，最多有$r-1$条边涉及$j$。

(f)通过将顶点1染成红色，将其他顶点染成红色或者蓝色，并在当且仅当u和v有相对颜色时设置u—v，可以得到2^{n-1}个完全有向图。由习题85(c)中的第一个式子知，对于每个这样的图的最小项B必须包含在这些项 $T = \delta_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \vee x_{k(i)}) = \lceil (G \cup H)^* \rceil \wedge \overline{\lceil G \cup H \rceil}$ 中的一个中。（例如，如果n=4且顶点(2,3,4)是(红，蓝，蓝)，那么 $B = \bar{x}_{12} \wedge x_{13} \wedge x_{14} \wedge x_{23} \wedge x_{24} \wedge \bar{x}_{34}$ 。）一个最小项B被包含在T中，当且仅当在B的染色过程中，$(G \cup H)^*$的一些边有不同颜色的顶点，但是$G \cup H$的所有边都是单色的。我们将证明T包含最多$2^{n-r}r^2$个这样的B。

设G为任意的图，且 $T = \lceil G^* \rceil \wedge \overline{\lceil G \rceil}$。下列的（效率较差）算法可以用来找到$G^*$：如果存在一个$r$组$S$有$|\Delta(S)|<2$，那么当$G^*=\infty$时停止。否则，如果$\Delta(S)=\{u,v\}$且$u \not— v$，那么将边$u$ —v加入到G并循环。

对于$1 \leqslant j \leqslant r$，当$|\Delta(S)|<2$时，最多$2^{n-r}$个两分的最小项$B$有单色的$\{u_j,v_j\}$。而且当$\Delta(S)=\{u,v\}$时存在$2^{n-r-1}$个有单色的$\{u_j, v_j\}$和双色的$\{u, v\}$。因此我们想要证明：当$G$是强$r$闭时，对于$G$的算法使用了少于$2r^2$次的迭代。

对于$k \geqslant 1$，设u_k—v_k是加入到G的第一条新边，它脱离自$\{u_j,v_j\}$，其中$1 \leqslant j<k$。由“强”的定义知，最多有r条这样的边存在；而且它们中的每一个都紧跟着最多$2r-3$条新边，这些边其中的一个端点是u_j或者v_j。因此找寻G^*的总步骤数最多是$r(2r-2)+1<2r^2$。

(g)习题84告诉我们 $q < \binom{p}{2} + (p+1)\binom{n}{2}$。因此我们要么有$2(r-1)^3 p \geqslant \binom{n}{3} - (r-1)^2 (n-2)$，要么有 $\binom{p}{2} + (p+1)\binom{n}{2} > 2^{r-1}/r^2$ 。当 $r = \left\lceil \lg\left(\frac{n^6}{186624(\lg n)^4}\right) \right\rceil$ 时，p的下界都是 $\frac{1}{12}\left(\frac{n}{6\lg n}\right)^3 \left(1 + O\left(\frac{\log\log n}{\log n}\right)\right)$。

[Noga Alon和Ravi B. Boppana，《Combinatorica》7 (1987), 1—22，在使用这种方法上取得了进展，除了别的以外，他们还证明对于任何单色链中的∧的数量的下界是$\Omega(n/\log n)^s$，这些任意的链决定G是否有一个固定大小$s \geqslant 3$的团。]

87. 当X是一个0-1矩阵时，X^3的条目最多是n^2条。一条有$O(n^{\lg 7}(\log n)^2)$个门的布尔链可以在整数模 $2^{\lfloor \lg n^2 \rfloor + 1}$ 上应用Strassen的矩阵相乘算法：4.6.4节的(36)。

88. 存在1 422 564个这样的函数，它们位于与变量的排列相关的716个类中。

算法L和这一节其他的方法很容易扩展成针对三元操作，而且我们为只针对最优中位数的计算得到下列的结果。

$C(f)$	类	函数	$C_m(f)$	类	函数	$L(f)$	类	函数	$D(f)$	类	函数
0	1	7	0	1	7	0	1	7	0	1	7
1	1	35	1	1	35	1	1	35	1	1	35
2	2	350	2	2	350	2	2	350	2	13	5670
3	9	3885	3	9	3885	3	8	3745	3	700	1416822
4	48	42483	4	48	42483	4	38	35203	4	1	30
5	201	406945	5	188	391384	5	139	270830	5	0	0
6	353	798686	6	253	622909	6	313	699377	6	0	0
7	99	169891	7	69	134337	7	176	367542	7	0	0
8	2	282	8	2	2520	8	34	43135	8	0	0
9	0	0	9	0	0	9	3	2310	9	0	0
10	0	0	10	0	0	10	0	0	10	0	0
11	0	0	∞	143	224654	11	1	30	11	0	0

S. Amarel-G. E. Cooke和R. O. Winder[《IEEE Trans.》EC-13 (1964), 4—13, 图5b]猜想这个具有9次操作的式子

$\langle x_1x_2x_3x_4x_5x_6x_7\rangle=\langle x_1\langle\langle x_2x_3x_5\rangle\langle x_2x_4x_6\rangle\langle x_3x_4x_7\rangle\rangle\langle\langle x_2x_5x_6\rangle\langle x_3x_5x_7\rangle\langle x_4x_6x_7\rangle\rangle\rangle$是通过3的中位数计算7的中位数的最好方式。但是这个有“魔力”的式子$\langle x_1\langle\langle x_2\langle x_3x_4x_5\rangle\langle x_3x_6x_7\rangle\rangle\langle x_4\langle x_2x_6x_7\rangle\langle x_3x_5\langle x_5x_6x_7\rangle\rangle\rangle\rangle\rangle$只需要8次操作，而且事实上最短的链只需要7步。

$$\langle x_1x_2x_3x_4x_5x_6x_7\rangle=\langle x_1\langle x_2\langle x_5x_6x_7\rangle\langle x_3\langle x_5x_6x_7\rangle x_4\rangle\rangle\langle x_5\langle x_2x_3x_4\rangle\langle x_6\langle x_2x_3x_4\rangle x_7\rangle\rangle\rangle$$

最有趣的函数$f(x_1,\cdots,x_7)=(x_1\wedge x_2\wedge x_4)\vee(x_2\wedge x_3\wedge x_5)\vee(x_3\wedge x_4\wedge x_6)\vee(x_4\wedge x_5\wedge x_7)\vee(x_5\wedge x_6\wedge x_1)\vee(x_6\wedge x_7\wedge x_2)\vee(x_7\wedge x_1\wedge x_3)$的质蕴涵项与投影面在7个点相关，它是所有的里面最难以置信的：它的最短长度$L(f)=11$和最小深度$D(f)=4$是通过这个非凡的式子

$$\langle\langle x_1x_4\langle x_4x_5x_6\rangle\rangle\langle x_3x_6\langle x_1\langle x_2x_3x_7\rangle\langle x_2x_5x_6\rangle\rangle\rangle\langle x_2x_7\langle x_1\langle x_5x_2x_4\rangle\langle x_5x_3x_7\rangle\rangle\rangle\rangle$$

得到的。

下面甚至更令人震惊的链以最优的方式计算它。

$$x_8=\langle x_1x_2x_3\rangle,\quad x_9=\langle x_1x_4x_6\rangle,\quad x_{10}=\langle x_1x_5x_8\rangle,\quad x_{11}=\langle x_2x_7x_8\rangle,$$

$$x_{12}=\langle x_3x_9x_{10}\rangle,\quad x_{13}=\langle x_4x_5x_{12}\rangle,\quad x_{14}=\langle x_6x_{11}x_{12}\rangle,\quad x_{15}=\langle x_7x_{13}x_{14}\rangle$$